Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Grau en Enginyeria en Tecnologies Aeroespacials

Title:

Study of optimization for vibration absorbing devices applied on airplane structural elements

Document content: REPORT

Delivery date: 27/06/2014

Author: Edgar Matas Hidalgo

Director: Meritxell Cusidó Roura
Codirector: Robert Arcos Villamarín

Mais la voix me console et dit: ≪Garde tes songes:
Les sages n'en ont pas d'aussi beaux que les fous!≫
Charles Baudelaire

Acknowledgements

I would like to thank all the people who have support this study.

First, I want to thank the directors of this study Meritxell Cusidó and Robert Arcos who have perform a really good follow-up work assisting me whenever I needed it.

I cannot forget the active role that Pierre Huguenet and Emiliano Tolosa from SENER Ingeniería y Sistemas had, without their advice and knowledge transfer this would have not been possible. I also want to recognize the paper of Vicenç Puig who has been the main advisor in the optimization field.

Especial thanks to my colleague Dídac Sellés who has work hand in hand with me. With the development of his own final degree study he has help this one to success.

Finally, I would like to show gratitude to my family and friends, they have stood by my side for years until arriving here.

Contents

List of Figures

List of Tables

1. Aim of the study

The aim of this study is to develop a process which allows to optimize the number, position or dynamic characteristics of the vibration absorbing devices that will be placed airplane structural elements in order to achieve a vibration reduction.

2. Scope of the study

This text studies a methodology to optimize the vibration absorber devices in a structure in order to minimize the vibration in some of its positions.

The final objective is to have a generic tool able to optimize a problem in where different vibration absorber devices (dynamic absorber devices, point masses and stiffeners) can be placed in a shell structure as an airplane fuselage.

To achieve this, the existing methodologies for the calculation of the dynamic response of a linear elastic structure are reviewed. Then, an alternative methodology to assemble the vibration reduction devices to the structure which does not need to recompute a Finite Element Model (FEM) for each of the possible configurations is applied. This methodology has been developed by E.G. Tolosa in [1] and D.Sellés in [2].

Once the methodology is known, the optimization problem is properly formulated and at least one optimization tool is developed to solve this problem.

Finally, each of the tools developed is tested in a practical case and its results are analyzed.

The development of the study is outlined in three different problems increasing its complexity: the fist problem concerns an unidimensional vibration reduction in a plane plate with dynamic vibration absorbers (DVAs) and point masses. Afterwards this problem is expanded to a 3D structure such as an airplane fuselage. And in the last case, stiffeners are considered as another possible vibration reduction device in a 3D structure.

3. Requirements

This study must fulfill the following requirements:

To develop a software computationally cheap enough to calculate the dynamic response of a pre-calculated structure coupled with a combination of vibration countermeasures (DVAs, point masses and stiffeners) faster than methods of re-calculation the whole system using, for example, FEM.
To develop an optimization cycle that allows time reduction in computation with respect to compute all the possible combinations.
To be able to deal with problems where different kind of devices can be placed in the structure.
To develop optimization tools flexible and simple enough to be adapted with small variations to a wide variety of cases.

4. Justification

Modern day aerospace engineering focuses a lot of effort on weight reduction and efficiency, as well as environmental impact reduction and comfort. Structural vibration generates unpleasant noises that contribute in the discomfort of passengers and attempting to solve vibration issues on aerospace structures often involves an increase in weight, which leads to fuel inefficiency.

Reducing noise and vibration is something in which a lot of resources have been spent, and it is especially difficult to achieve a significant reduction in the final stages of the design because it usually involves great changes in multiple aspects of the vehicle.

However, one of the possible methods is performing a passive vibration control adding some vibration absorbers to the structure such as DVAs, point masses or stiffeners. This is very useful in a complex structural system such as an airplane fuselage because it potentially improves its dynamic response without making important changes in the whole structural system. However, current methods of vibration absorption employing these devices consist mostly on the study of several configurations, placed on the geometry of the structure following the engineer's experience. This method becomes more precise as the engineer analyses more and more cases.

An exhaustive method of comparison, if performed experimentally, is too expensive and time consuming. If the amount of data to be processed is high enough, a method of all cases comparison is also too computationally costly to be performed via simulations. This study develops a software to be used specifically in an optimization cycle, containing an algorithm capable of discarding most far-to-optimal cases, which potentially reduces the time in such a way as to be able to numerically find the best configuration of vibration absorbing devices set on a certain system, amongst a wide variety of combinations.

5. State of the art

Since vibration control is a key aspect in many engineering fields, several investigation teams and private corporations are focusing their efforts on improvement of vibration behavior of structures in the last stages of its design. In particular, the aerospace industry is aiming for an improvement in weight and functionality of the aircraft structures by optimizing their vibration response. In general, vibration improvements are being studied with the help of experts that provide qualitative criteria born of years of experience.

With the progressive increase of computational power, companies and research groups are recently focusing on numerical methods for vibration optimization, rather than trial and error and expert guessing.

One such company is the Spanish SENER Ingeniería y Sistemas, with the development of formulation, simulations and testing of optimization methods for vibration countermeasures [3], and partner to this study.

Efforts have been directed toward the improvement of DVA architecture, in order to avoid internal resonance [4], or to completely redesign the architecture of a DVA [5]. Other studies focus on an in-depth study of the effect of a single DVA in a large scale of frequencies, evaluating the importance of each of the characteristic parameters of the device [6].

Other approaches consist on the development of methods of optimization with genetic algorithms focusing on parameters of active vibration control [7], and others have used Non Nominated Genetic Algorithms to optimize the characteristics of the DVAs themselves [8].

As it is seen, there is no news of a study with a similar aim to the one developed in this text. However, that does not means that it hasn't been developed for some company but the hight industrial interest of this methodology could make it difficult to found references.

6. Introduction to the problem and theoretical basis

The problem that is solved in this study consists in finding the optimal combination of devices (DVAs, point masses or beams), considering optimal the one that minimize the vibration is a set of points that will be given by the user (objective points from now on) while fulfilling the next limitations:

- The different devices only can be placed in a finite set of points that will be given as input data of the problem.
- Only one device can be placed in each of the points given.
- There could be different types of devices to place in the structure. There can be conceptually different devices (DVAs, point masses or beams) but inside each of these groups different types of devices can be differentiated by their physical properties. The set of properties defining each type of device will be given by the user.
- There will be a limitation in the number of devices that can be placed in the structured, also defined by the user.

Thus, to achieve this objective the problem is solved in different stages increasing its complexity.

The first stage solves the problem just placing DVAs and point masses (not beams) in a plane plate. The theoretical basis are explained in section 6.1.

In the second stage, the problem becomes three-dimensional and DVAs and point masses are placed in a shell structure, specifically in an airplane fuselage. Theoretical basis of this problem is in section 6.2.

The last stage introduce beams to the previous problem. Then, a variety of DVAs, point masses and beams can be placed in an airplane fuselage, or in general in a shell structure, as to reduce the vibration level. The theory related with beams is introduced in section 6.3.

6.1 Theoretical basis of the DVAs in a plate problem

The problem of placing DVAs is based in the so called Equivalen Force Method developed by E.Tolosa in collaboration with SENER Ingeniería y Sistemas SA in [1] which is a direct application of the receptance method presented in [9].

This method propose an equivalent force applied in a certain point i which has the same effect on the structure that a particular DVA placed there. This equivalent force is a function of the working frequency (ω), the three physical characteristics that defines the DVA: mass (m_i) , spring stiffness (k_i) and coefficient of viscous damping (c_i), and also the amplitude of the vibration in the application point (x_i).

$$f_d^i = \frac{\omega^2(k_i + c_i\omega j)m_i}{k_i - m_i\omega^2 + c_i\omega j}\, x_i \tag{6.1}$$

By means of an analogy between this equivalent force and the force generated with a single mass ($f = \omega^2\, m\, x$) [9], it can be defined the so called equivalent mass of the DVA.

$$m_{eq}^i = \frac{(k_i + c_i\omega j)m_i}{k_i - m_i\omega^2 + c_i\omega j} \tag{6.2}$$

This result can be expressed in matrix form as

$$\underline{\mathbf{F}}_i^d = \omega^2 \begin{pmatrix} m_{eq_1} & 0 & 0 & \dots & 0 \\ 0 & m_{eq_2} & \dots & \dots & \\ 0 & \vdots & \ddots & & \\ \vdots & \vdots & & & \\ 0 & & & & m_{eq_n} \end{pmatrix} \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ \\ x^n \end{pmatrix} = \mathbf{a}\underline{\mathbf{x}} \tag{6.3}$$

where the bold underlined variables denote vectors, n is the total amount of points in the problem (both objective and possible points) and the terms m_{eq_i} are the equivalent masses in the point i computed by equation 6.2. The matrix $\mathbf{a}$ is defined as the matrix containing the information of a given combination of DVAs.

Applying this force (equation 6.3) to the plate structure it is obtained the linear system of equations that allows to compute the vibration of each point of the problem for a given combination of DVAs:

$$(\mathbf{I} - \mathbf{H}\,\mathbf{a})\,\underline{\mathbf{x}} = \underline{\mathbf{x}}_0 \tag{6.4}$$

This relationship has been build by means of the Frequency Response Function matrix ($\mathbf{H}$ or FRF matrix from now on). As this is a unidimensional problem and only the vibration

in the perpendicular direction of the plate is considered, each point corresponds to a single degree of freedom (DOF). Then, each term in this matrix (H_{ij}) represents the displacement produced in the point i if a unit force is applied in point j.[1]

The term $\mathbf{I}$ represents the identity matrix and the vector $\underline{\mathbf{x}}_0$ is the vibration of the structure when there is no device placed. This term can be computed as a the product of the FRF matrix and the vector defining the external force applied to the structure ($\underline{\mathbf{F}}$)

$$\underline{\mathbf{x}}_0 = \mathbf{H}\,\underline{\mathbf{F}}. \tag{6.5}$$

Regarding the point masses, they can be treated as a particular case of a DVA with an infinity value of spring stiffness ($k_i \to +\infty$). It can be proved by computing the limit of equation 6.2

$$\lim_{k_i \to +\infty} m_{eq}^i = \lim_{k_i \to +\infty} \frac{(k_i + c_i\omega j)m_i}{k_i - m_i\omega^2 + c_i\omega j} = m_i, \tag{6.6}$$

which is obviously the expected result for a point mass.

Therefore, from now on there will be no discernment between DVAs and point masses but their physical properties, and all what is explained to DVAs can be extended to point masses.

6.2 Theoretical basis of the DVAs in a fuselage problem

The problem of placing DVAs in a fuselage or a general shell structure is the natural 3D extension of DVA in a plane plate problem (section 6.1). It has been studied in collaboration with D.Sellés for his Degree thesis (June 2014).

To formulate the problem in three dimensions, each equivalent mass becomes a matrix that expressed in local coordinates (x' axis coinciding with the axis of the DVA) have only one non-zero term corresponding with the equivalent mass defined in equation 6.2.

$$M_{eq}^{i'} = \begin{pmatrix} m_{eq}^i & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{6.7}$$

Where the prime notation indicates global coordinates.

This matrix has to be rotated in order to be expressed in global coordinates and ensemble all the matrices from different points. The rotation can be expressed in matrix notation as:

[1] The full development of this formulation can be found in [1].

$$\mathbf{M}_{eq}^{i} = \mathbf{R}^{t}\ \mathbf{M}_{eq}^{i'}\ \mathbf{R} \tag{6.8}$$

The rotation matrix ($\mathbf{R}$) is the expression for the rotation in global coordinates, and it has the following structure:

$$\mathbf{R} = \begin{pmatrix} \cos\phi\cos\psi & \sin\phi\sin\theta - \cos\phi\sin\psi\cos\theta & \cos\phi\sin\psi\sin\theta + \sin\phi\cos\theta \\ \sin\psi & \cos\psi\cos\theta & -\cos\psi\sin\theta \\ -\sin\phi\cos\psi & \sin\phi\sin\psi\cos\theta + \cos\phi\sin\theta & \cos\phi\cos\theta - \sin\phi\sin\psi\sin\theta \end{pmatrix} \tag{6.9}$$

Where the angles θ, ϕ, ψ represent the rotation of the local axis with respect to the *x*, *y* and *z* axis of the global coordinates.

Once the equivalent mass matrices for the DVA elements are expressed in global coordinates, matrix $\mathbf{a}$ from equation 6.3 can be rewritten as the direct summation of the sub-matrices generated in equation 6.8 :

$$\mathbf{A} = \omega^2 \bigoplus_{i=1}^{n} \mathbf{M}_{eq}^{i} \tag{6.10}$$

Where n is the total amount of points of interest in the problem.

Once matrix $\mathbf{A}$ is defined, equation 6.4 can be extended to the three space dimensions:

$$(\mathbf{I} - \mathbf{H}\ \mathbf{A})\ \underline{\mathbf{X}} = \underline{\mathbf{X}}_0 \tag{6.11}$$

All the parameters in this equation have the same meaning that in equation 6.4 but they have been extended to the three space dimensions. For each point of interest every one of these vectors and matrices contains three terms, corresponding with the components

x, y, z. As example, the vector $\underline{\mathbf{X}}$ will look like:

$$\underline{\mathbf{X}} = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \\ \vdots \\ \vdots \\ x_n \\ y_n \\ z_n \end{bmatrix} \tag{6.12}$$

6.3 Theoretical basis of the beams in a fuselage problem

Once the problem is extended to the space, beams are included to the structure. This theory has been developed by D.Sellés for his Degree thesis (June 2014). The beams are considered undamped Euler's beams.

Similarly to what has been done in section 6.1 it can be found a equivalent force that has the same effect in the structure that the corresponding beam. However, when dealing with a beam, the equivalent force will affect the two points where it is in contact with the structure. This force is given by the expression

$$\underline{\mathbf{F}}_b = (\mathbf{K} - \omega^2 \mathbf{M})\, \underline{\mathbf{x}}' \tag{6.13}$$

where

- $\underline{\mathbf{F}}_b$ is the vector containing forces and torques applied on the nodes of a beam element.
- $\underline{\mathbf{x}}'$ is the vector containing displacements and twists of the nodes of a beam element.
- $\mathbf{K}$ is the elemental stiffness matrix.
- $\mathbf{M}$ is the elemental mass matrix.

The elemental stiffness matrix is a function of the following properties of the beam:

- Longitudinal elasticity modulus (E).
- Transversal elasticity modulus (G).
- Moments of inertia (I_x, I_y, I_z).
- Cross section area (A).
- Beam length (L).
- Torsion constant (J).

and is defined in the annex A.

While the elemental mass matrix is function of the:

- Material density (ρ).
- Moments of inertia (I_x, I_y, I_z).
- Cross section area (A).
- Beam length (L).

and is also defined in the annex A.

In order to obtain a expression in the form of equation 6.13 valid for a combination of beams placed in a three-dimensional structure, both $\mathbf{M}$ and $\mathbf{K}$ elemental matrices have to be expressed in global coordinates, and assembled

$$\underline{\mathbf{F}} = (\mathbf{K}_G - \omega^2 \mathbf{M}_G)\, \underline{\mathbf{x}} = \mathbf{H}_B\, \underline{\mathbf{x}}, \tag{6.14}$$

where

- $\underline{\mathbf{F}}$ is the vector containing forces and torques applied on all the points of interest in the problem.
- $\underline{\mathbf{x}}$ is the vector containing displacements and twists of all the points of interest in the problem.
- $\mathbf{K}_G$ is the global stiffness matrix, result of the rotation and assembling of all the elemental stiffness matrices.
- $\mathbf{M}_G$ is the global mass matrix, result of the rotation and assembling of all the elemental mass matrices.

Notice that all these vectors and arrays contain six elements for point of interest.

Applying this force (equation 6.14) to the three-dimensional structure a linear equations system that allows to compute the vibration of each point of interest:

$$\left(\mathbf{I} + \mathbf{H}\left(\mathbf{K}_G - \omega^2 \mathbf{M}_G\right)\right) \underline{\mathbf{x}} = \underline{\mathbf{x}}_0. \tag{6.15}$$

For the sake of simplicity, let it be

$$\mathbf{H}_B = \left(\mathbf{K}_G - \omega^2 \mathbf{M}_G\right) \tag{6.16}$$

from now on. The last step to achieve the combination of DVAs and beams in a structure is to combine equation 6.15 with the one deduced for DVAs (equation 6.11):

$$\left(\mathbf{I} + \mathbf{H}\left(\mathbf{H}_B - \mathbf{A}\right)\right) \underline{\mathbf{x}} = \underline{\mathbf{x}}_0. \tag{6.17}$$

Despite the abuse of notation, notice that equation 6.11 does not take into account the moments or twists. Then, in order to ensure the dimensional concordance between matrices, the $\mathbf{A}$ matrix from equation 6.11 must be extended by introducing zeros in those rows or columns corresponding with the twists in the structure.

The full rigorous deduction of this equation can be followed at [2].

6.4 Objective discussion

The objective of the problem is to minimize the vibration of some of the points in the structure, the so called objective points.

The most common way of expressing this reduction in the bibliography is the insertion loss (IL). This magnitude represents the attenuation of the vibration produced by the insertion of the devices in decibels (dB) as:

$$\mathsf{IL} = 20 \log \left(\frac{\|\underline{\mathbf{x}}_0\|}{\|\underline{\mathbf{x}}\|} \right), \tag{6.18}$$

where $\|\underline{\mathbf{x}}_0\|$ is the modulus of the vibration vector which contains only the objective points when there is no device in the structure (initial problem) and $\|\underline{\mathbf{x}}\|$ is the modulus of the vibration vector with the current configuration of devices placed in the structure. They are

computed as:

$$\|\underline{\mathbf{x}}_0\| = \sum_{i=1}^{m_o} \left(\sqrt{\mathrm{Re}(x_{0_i})^2 + \mathrm{Im}(x_{0_i})^2} \right) \tag{6.19}$$

$$\|\underline{\mathbf{x}}\| = \sum_{i=1}^{m_o} \left(\sqrt{\mathrm{Re}(x_i)^2 + \mathrm{Im}(x_i)^2} \right). \tag{6.20}$$

where m_o is the total amount of objective points.

The insertion loss expresses the reduction of vibration with positives values and the increase with negative values, both in dB. Therefore, the objective is to achieve the higher IL value possible.

This way of expressing the reduction is very practical to give final results and to be interpreted by engineers, however, this function is quite complex and it is not a good candidate to be used as objective function in an optimization process.

Analyzing the IL function it is easily seen that maximizing its value is equivalent to minimizing the modulus of the vibration vector for the current configuration ($\|\underline{\mathbf{x}}\|$). Moreover, considering that this is not a continuous problem but a discrete one because the solution must be contained in a set of possible configurations, it is that, obviously, there cannot be a fraction of device placed in the structure, this function can be simplified even more. The function suggested is the quadratic function

$$\Psi = \sum_{i=1}^{m_o} \left(\mathrm{Re}(x_i)^2 + \mathrm{Im}(x_i)^2 \right). \tag{6.21}$$

Although minimizing equation 6.21 is not formally equal to minimizing equation 6.20, considering the discrete nature of the problem, the error is negligible and will not affect the optimal combination of devices. Furthermore, using a quadratic function as a objective function provides great benefits for optimization (see chapter 7).

In order to prove this, a simple problem where all the possible combination can be computed and compared is analyzed. The case consist in a plane plate where 9 possible points have been chosen. Only one type of DVA can be placed in it and a force is applied at a frequency of $320Hz$. The properties of the plate and the DVA can be found in annex Bbut are not relevant for this discussion.

This case, has

$$2^9 = 512$$

possible combinations of DVAs, and the system from equation 6.4 is solved for all of them. These combinations are represented by 9 digits binary numbers. A 0 means that there is not device in that position and a 1 that there is a device placed there. This combinations

are coded by their equivalent decimal number in the graphs.

In figure 6.1 it is plotted the objective value by the modulus of the vibration (equation 6.20) for all the possibilities.

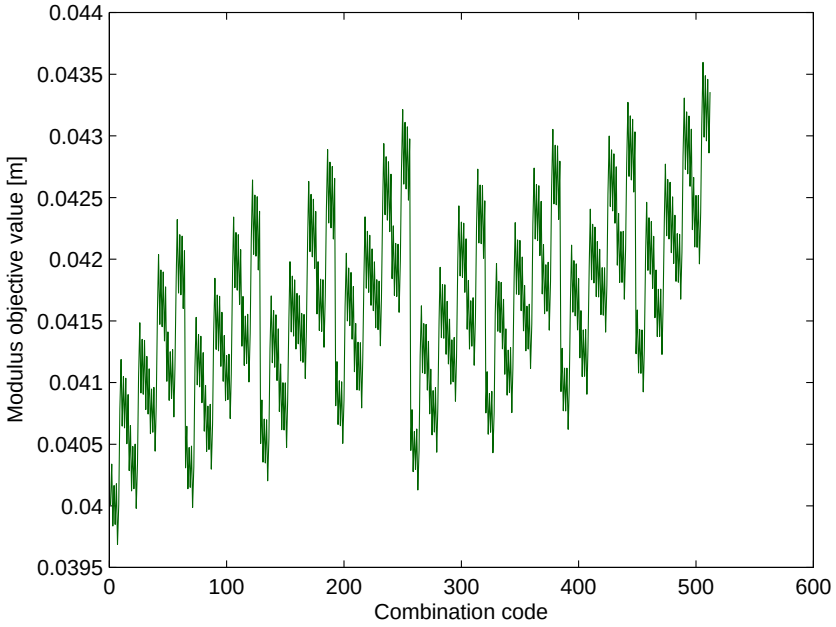

Figure 6.1: Objective value computed with the modulus function for all the possible combinations.

And in figure 6.2 it is plotted the objective valued computed by equation 6.21.

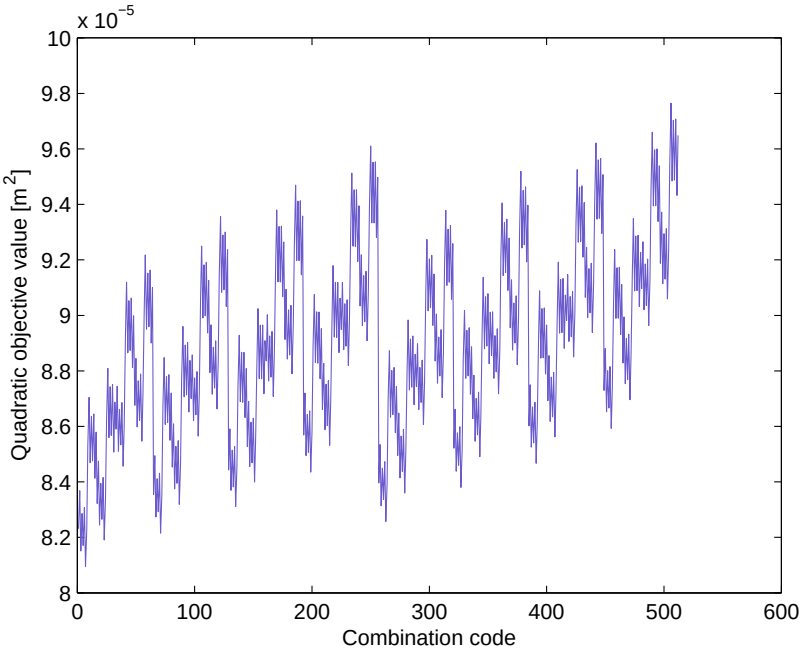

Figure 6.2: Objective value computed with the quadratic function for all the possible combinations.

It is clear that the value of the objective is not the same, but the minimum is in the first combinations for both of them. In figure 6.3 and figure 6.4, it is zoomed into this area and it observed that the minimum vibration in given for combination 6 in both of them. Which means that the minimum vibration is achieved placing two devices, one in position number 7 and the other one in position number 8 (combination code 000000110).

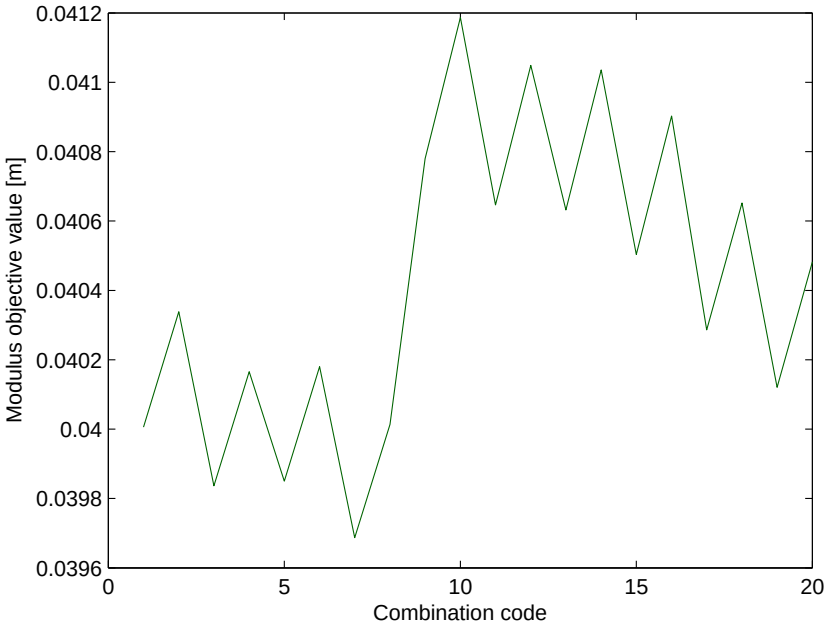

Figure 6.3: Zoom into the minimum objective value computed with the modulus function.

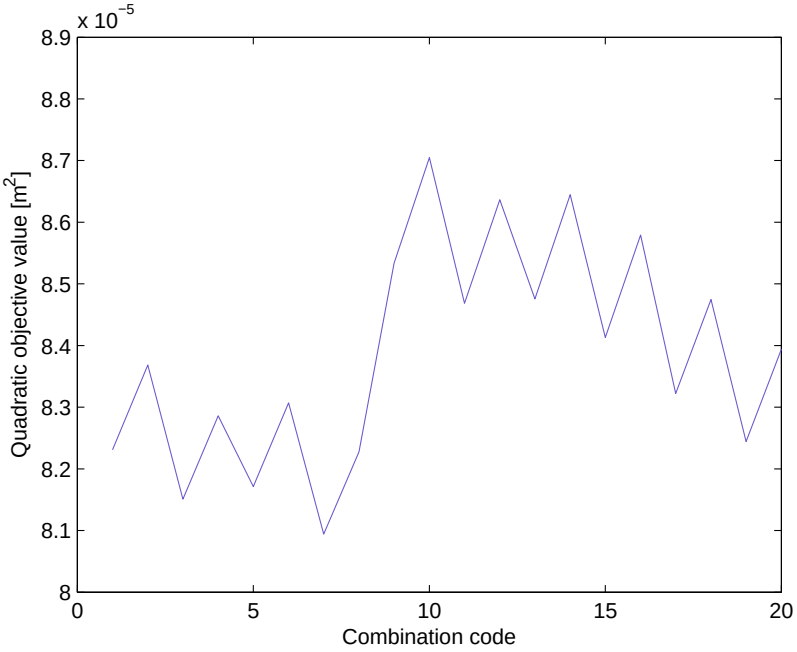

Figure 6.4: Zoom into the minimum objective value computed with the quadratic function.

It is proved then that the error is negligible and from now on the objective of the problem will be to minimize the result of equation 6.21.

7. Theoretical introduction to optimization

Although the problems presented in section 6 hugely reduced the computational time required to evaluate each combination with respect to the FEM simulation alternative, solving the problem for all the possible combinations is absolutely infeasible when dealing with a real problem. It is because the number of combinations increase exponentially for the number of possible points where a device can be placed (m_p) as

$$C = 2^{m_p} \tag{7.1}$$

for a single device, and they increase even more when it is possible to chose between more than one device becoming of the order of

$$C = (t+1)^{m_p}, \tag{7.2}$$

where t is the amount of different kinds of devices that can be placed in the structure [1].

Therefore, it is needed to formulate it as an optimization problem in order to be solved.

It is important to remark that this problem has an added difficulty as it is not a continuous problem. A device can be placed or not but it cannot have partial effect in the structure because it will have no physical sense.

Taking into account the characteristics of the problems presented in the previous section and thinking in the size of the cases of real interest as can be the structure of an airplane, two alternatives are presented in order to solve the problem

- Mixed-Integer Quadratic Programming (MIQP) and
- Genetic Algorithms (GA).

A brief introduction to both of this options is presented in the next sections besides its main strong points and the reason for which they have been chose.

[1] It is not exactly this amount because the number of devices that can be placed in the structure is bounded below m_p.

7.1 Mixed-Integer Quadratic Programming

Mixed-Integer Quadratic Programming (MIQP) is an extension of linear optimization programming (LP)[10], which is a method to achieve the best outcome in a mathematical model whose constraints are linear relationships. A mathematical model can be expressed by:

- An **objective function**,
- a set of **constraints** and
- upper and lower **bounds** for the variables.

In LP, the constraints must be **linear** relationships[11]. And the quadratic extension allows the objective function to include quadratic terms, while the mixed-integer extension allows some of the variables to be determined to be restricted to an **integer** value. In particular, the problem to solve is a binary problem because the integer variables only can achieve the values 0 or 1, in order to represent if a device is placed or not.

MIQP is the most formal approach that can be done in a problem as those presented in section 6.1 and section 6.2 where the constraints can be linearized (see section 9.1.2 and section 10.1.2). However in the problem presented in section 6.3 this methodology cannot be used because the beams representing the stiffeners connect two different points of the structure, then the matrices $\mathbf{K}_G$ and $\mathbf{M}_G$ (equation 6.15) have to be build by an assembling taking into account which beams have to be placed and which have not and this process cannot be expressed as a set of linear relationships.

The strong points of this formulation is that the algorithms used to solve the problem (cutting planes, branch and bound...) are exact algorithms which ensures to find a global optimum [12]. Moreover, huge efforts have been focused in developing this optimization methods and there are optimization tools as IBM ILOG CPLEX [13] which solves the problem with great efficiency.

On the other hand, MIQP presents problems of scalability. Huge problems may be intractable in terms of solving time as the number of combinations rapidly increase as seen in equation 7.1 which makes infeasible to ensure that the solution found is a pure optimum in a reasonable time [14]. Therefore, for dealing with these problems and to include stiffeners in the structure an heuristic alternative algorithm is needed.

7.2 Genetic Algorithms

The other alternative to solve the problem is genetic algorithms (GA). These are search heuristics algorithms inspired in Darwin's theory of evolution.

In GA there is a **population** made of different **individuals** which contain the coded information about the solution they represent. These individuals belong to the search space defined by the constraints of the problem.

The objective function in these problems is evaluated for the different individuals in the population, its value is known as **fitness**. Then, the aim of the algorithm is to achieve the best possible fitness (maximum or minimum depending on the problem).

To achieve that, in every iteration or **generation** of the algorithm different strategies are performed in order to take advantage of the current individuals, which fitness is known. These strategies can combine different individuals (**procreation**), randomly change some pieces of code from an individual (**mutation**), coping the best individuals to the next generation (**elitism**) or even generating new random individuals. These strategies make the algorithm learn from the previous iterations and this works to improve the best individual found instead of performing a simple search and trying only random solutions [12].

The main application of genetic algorithm are problems like the ones studied in this text for which it is *fast* to check a solution but it is very *difficult* to find the optimal as the number of possible combinations is huge (of the order of 2^n)[2]. For these problems, the time required to solve them with exact algorithms is incredibly huge (or even infinite) [15].

Another advantage of GA is that they don't need the constraints to be linear or to have any specific mathematical property, besides they are easy to implement and to adapt to different problems. This will allow to solve the problem presented in section 6.3 that cannot be formulated in MIQP.

On the other hand, GA never ensure that a global optimum has been reached and they may take more time to find it than MIQP but with a relatively low level of effort a good solution can be found in a relatively low amount of time.

[2]They are NP (nondeterministic polynomial time) problems but the explanation of these concepts are out of the scope of this study. For further information see [16].

8. Software and programming languages

This section offers an explanation of the reasoning used for the choice of software used in different parts of the project. The programming language used for the preprocessing of the data is also justified. The choice in this case has been Python 2.7.

There are three main items through the project in need of specific software.

- **Genetic algorithm:** The problem to solve is a matrix problem, and the genetic algorithm demands a choice of platform to be implemented in. The software chosen in this case has been MATLAB 2013.
- **MIQP algorithm:** Just as in the genetic algorithm, a choice of platform for the MIQP algorithm is needed, but the implementation of the optimization process must be integrated with a MIQP solver. The software chosen in this case is the CPLEX Python API.
- **Frequency response function calculation:** The process needs the FRF matrix of the structure under evaluation as an input. The choice of software for this task has been MD Nastran 2010

8.1 MATLAB 2013

MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming, developed by the American computation software company MathWorks.
The choice of MATLAB as the platform is its spread usage in all fields of engineering, along with its ability to process and use text files as inputs. This programming language is specially confortable to deal with matrix problem and allows the engineers to greatly reduce the amount of time required to develop a program. The experience of the author working with this language and its accessibility through university licenses are also a key factor for the choice of this software. Since MATLAB is well known to engineers all over the world, a user can spend little time modifying the code designed in this project to suit variations in its use.

8.2 CPLEX Python API

IBM ILOG CPLEX is an optimization software package working mainly on a base of C language, although recent versions also support other bases. The interfaces available for CPLEX are in a variety of programming languages. CPLEX was originally developed by Robert E. Bixby and later acquired and expanded by the American software company IBM.
The software has been chosen for its programming efficiency, the inclusion of an MIQP algorithm and its recognition among the academic community. The experience with this optimization tool of some of the people who have support this study and the availability of free licenses for academic use have made the decision clear.

CPLEX solvers can be called from different programming languages, the CPLEX Python API has been chosen since Python it is also the language of choice for the preprocessor. The version used was 12.6 as it is the lasted version to date.

8.3 Python 2.7.6

Python is a high-level programming language with a syntax allowing the user to express commands in fewer code lines than other choices such as C. Python can act both as a shell or as a scripting method. Its main versions are implemented on a C base, so most of its libraries are written in C language.
The main need of the project regarding a vehicular programming language is the text file generation and modification, so a short and versatile functionality for these actions has been the main reason for choosing Python over, for example, C or Java. Python includes a big amount of functions regarding search, modification, combination and multiplication of lines, chains of characters and whole text files in a very compact notation. Besides, it is much simple and clear than traditional languages as C, which greatly reduce the program developing time.

Python 2.7.6 has been chosen because it is the latest version compatible with CPLEX Python API.

8.4 MD Nastran 2010

The vibration response to harmonic forces of the model is calculated with **MD NASTRAN 2010**. NASTRAN, the NASA Structural Analysis System, is a FEA (Finite element analysis) software originally developed by NASA and improved and commercialized by the MacNeal-Schwendler Corporation (MSC), as MSC NASTRAN, with a further version

called MD NASTRAN, which included more options regarding non-linear solution types. The software allows the user to access a wide variety of static and dynamic analyses, with a big choice of boundary conditions, meshing options, element types etc.
The choice of MD Nastran as the tool for generating the frequency response of a structure has been mainly due to the fact that Nastran is the referent software for vibration calculations and is widely recognized and used in both industrial and academical environments, and its outputs can be read and modified in an ASCII interpreter, allowing a great flexibility in their manipulation. The last observation is crucial to this project, since there has been a need to design a bridge between the Nastran outputs, Python and MATLAB data type.

9. DVAs in a plate

In this chapter it is discussed the optimization of the problem introduced in section 6.1 which aim is to find the best configuration of DVAs and point masses in a plane plate.

The problem is formulated in section 9.1, then the implementation of the optimization tools by GA and CPLEX solvers is explained in section 9.2 and section 9.3 respectively. Finally, in section 9.4, a practical case is solved and the results are analyzed.

9.1 Problem formulation

There is a structural system (a plate in this case). This system is meshed and two different sets of points are defined by their nodal number:

Objective points: set of points in which the vibration will be minimized.

$$\{Objective\ points\} = \mathcal{O} \in \mathbb{N}, \tag{9.1}$$

which size is m_o.

Absorber points: set of points in which an absorber can be placed.

$$\{Absorber\ points\} = \mathcal{D} \in \mathbb{N}, \tag{9.2}$$

which size is m_p.

There are also absorber devices (DVAs and point masses in this case). Each device can be of a different type. The device type will be coded as a natural number belonging to a finite set. This set is defined as:

$$\{Types\} = \mathcal{T} \in \mathbb{N}, \tag{9.3}$$

a set of size t.

A given configuration will be defined by:

- The amount of absorber devices placed in the structure:

$$n_d \in \mathbb{N} \quad ; \quad n_d \leq n_{d_{max}}. \tag{9.4}$$

- For each device there will be:
  - a variable to define the type

$$t_i \in \mathcal{T} \quad for \quad i = 1, ..., n_d \tag{9.5}$$

  - a variable to define the position

$$p_i \in \mathcal{D} \quad for \quad i = 1, ..., n_d. \tag{9.6}$$

Lets define the set of points in which there will be an absorber device as:

$$\mathcal{J} = \{p_i \mid i = 1, ..., n_d\} \quad \mathcal{J} \subset \mathcal{D}. \tag{9.7}$$

The **objective** of the problem is to minimize the vibration on the objective points ($\mathcal{O}$). As it has been arranged in section 6.4, this vibration will be represented by the quadratic sum of both real and imaginary part of each point in $\mathcal{O}$

$$\text{min}\Psi = \text{min}\{\sum_{j=1}^{m_o} (Re(x_j)^2 + Im(x_j)^2)\}, \tag{9.8}$$

where the vector $\underline{\mathbf{x}}$ is compute solving the system of linear equations presented in equation 6.4 in section 6.1:

$$(\mathbf{I} - \mathbf{H}\ \mathbf{a})\ \underline{\mathbf{x}} = \underline{\mathbf{x}}_0, \tag{9.9}$$

where $\mathbf{I}$ is an identity matrix of dimension $m_o + m_p$,

$H_{ij} \in \mathbb{C}$ for $i, j = \{1, ..., m_o + m_p\}$ are the terms of the FRFs matrix,

$x_{0_i} \in \mathbb{C}$ for $i = \{1, ..., m_o + m_p\}$ is the vibration of the point i when no device is placed in the structure,

$x_i \in \mathbb{C}$ for $i = \{1, ..., m_o + m_p\}$ is the unknown vibration of the point i with the current device configuration,

and the matrix $\mathbf{a}$ is a diagonal matrix defined by the equivalent masses of the device located at that point times the square frequency

$$\mathbf{a} = w^2 \begin{pmatrix} 0 & & & & & \\ & \ddots & & & & \\ & & 0 & & & \\ & & & m_{eq_1} & & \\ & & & & \ddots & \\ & & & & & m_{eq_p} \end{pmatrix}_{(m_o+m_p)\times(m_o+m_p)} . \tag{9.10}$$

The equivalent masses are a well known function of the type of absorber ($m_{eq_i} = f(t_i)$ for $i \in \mathcal{J}$) defined in equation 6.2.

Notice that for those positions belonging to $\mathcal{D}$ where there isn't a device,m_{eq_i} will be null as well as for those positions included in $\mathcal{O}$ but not in $\mathcal{D}$.

9.1.1 Genetic algorithm approach

To solve this problem with GA an individual which codifies the information about the devices placed in the structure has to be defined.

The more obvious alternative is to define a binary variable for each possible device but this is not the most efficient solution in terms of the space required (memory usage) because the number of variables needed ($m_p\ t$) can be much larger than the number of maximum devices placed ($n_{d_{max}}$).

Therefore, a more efficient codification is used. It consist in a matrix ($\mathbf{T}$) of dimensions $n_d \times 2$ which rows represents a device placed in the structure, the first column will contain the values of the variables t_i and the second one the values of p_i :

$$\mathbf{T} = \begin{pmatrix} t_1 & p_1 \\ t_2 & p_2 \\ \vdots & \vdots \\ t_{n_d} & p_{n_d} \end{pmatrix} . \tag{9.11}$$

This alternative obviously reduces the space required to save the algorithm but also the time required to decode it for a general problem as there is less data to treat.

9.1.2 MIQP formulation

The other alternative considered is MIQP. This methodology will use IBM ILOG CPLEX to solve the problem, it needs the problem to be defined by a set of linear constraints and

an objective function and requires to define some decision variables representing if a device is placed in the structure or nor, as explained in section 7.1. Therefore, the problem formulation is adapted in this section.

Decision variables

The decision variables will be a set of $m_o + m_p$ binary elements for each type of device ($\underline{\mathbf{z}^\mathbf{i}}$). Hence there will be t sets.

$$(\underline{\mathbf{z}^\mathbf{1}}, \underline{\mathbf{z}^\mathbf{2}}, ..., \underline{\mathbf{z}^\mathbf{i}} ..., \underline{\mathbf{z}^\mathbf{t}}) \; ; \;\; \underline{\mathbf{z}^\mathbf{i}} = \begin{pmatrix} z_1^i \\ z_2^i \\ \vdots \\ z_{(m_o+m_p)}^i \end{pmatrix} \; ; \; z_j^i \in \{0, 1\}. \tag{9.12}$$

Objective function

The objective is to minimize the addition of the square vibration amplitude in the objective points ($\mathcal{O}$):

$$\text{min}\Psi = \text{min}\{\sum_{j=1}^{m_o} \left(\text{Re}(x_j)^2 + \text{Im}(x_j)^2\right)\}. \tag{9.13}$$

Constraints

The constraints that build the problem are:

- Dynamic relationship of the structure. It has to be reformulated in order to introduce the decision variables with the desired effect.

  To achieve this, the matrix $\mathbf{a}^i$ is defined as a constant diagonal matrix

$$\mathbf{a}^i = \omega^2 \begin{pmatrix} m_{eq}^i & & & & \\ & m_{eq}^i & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & m_{eq}^i \end{pmatrix}_{(m_o+m_p)\times(m_o+m_p)} \tag{9.14}$$

  for each type $i \in \mathcal{T}$, and the matrix $\mathbf{z}^i$ is the conversion of vector $\underline{\mathbf{z}}^i$ in a diagonal matrix. Therefore, the matrix $\mathbf{a}$ can be build by the sum of the product $\mathbf{a}^i \; \mathbf{z}^i$ for each type $i \in \mathcal{T}$ and substituted in equation 9.9, yielding

$$\left(\mathbf{I} - \mathbf{H}\left(\sum_{i=1}^{t} \mathbf{a}^i\ \mathbf{z}^i\right)\right)\ \underline{\mathbf{x}} = \underline{\mathbf{x}}_0. \tag{9.15}$$

- Maximum one device per position belonging to the set of possible points ($\mathcal{D}$):

$$\sum_{i=1}^{t} z_p^i \leq 1\ \ \forall p \in \mathcal{D}. \tag{9.16}$$

- No devices in the positions belonging to the objective set of points ($\mathcal{O}$):

$$\sum_{i=1}^{t} z_p^i = 0\ \ \forall p \in \mathcal{O}. \tag{9.17}$$

- Maximum amount of devices ($n_{d_{max}}$):

$$\sum_{i=1}^{t}\sum_{j=1}^{m_p} z_j^i \leq n_{d_{max}}. \tag{9.18}$$

It's easily seen that all the constraint equations fulfill the requirements to be implemented in CPLEX but the dynamic relationship. In equation 9.15 two problems are noticeable:

- It contains variables and coefficient in the complex plane, and the solver can just work with reals or integers.
- The equation is not linear as there appears the product $\underline{\mathbf{z}^\mathbf{i}}\ \underline{\mathbf{x}}$.

To solve these problems, the first step is to divide the complex variables in two real variables, one representing the real part and the other the imaginary part. Hence:

$$\mathbf{H} = \mathbf{H_R} + \mathbf{H_I}j, \tag{9.19}$$

$$\mathbf{a^i} = \mathbf{a_R^i} + \mathbf{a_I^i}j, \tag{9.20}$$

$$\underline{\mathbf{x}} = \underline{\mathbf{x}}_R + \underline{\mathbf{x}}_I j, \tag{9.21}$$

$$\underline{\mathbf{x}}_0 = \underline{\mathbf{x}}_{0_R} + \underline{\mathbf{x}}_{0_I} j. \tag{9.22}$$

Introducing this new terms to equation 9.15:

$$\left(\mathbf{I} - (\mathbf{H_R} + \mathbf{H_I}j)\ \left(\sum_{i=1}^{t}\left(\mathbf{a_R^i} + \mathbf{a_I^i}j\right)\ \mathbf{z^i}\right)\right)\ (\underline{\mathbf{x}}_R + \underline{\mathbf{x}}_I j) = \left(\underline{\mathbf{x}}_{0_R} + \underline{\mathbf{x}}_{0_I} j\right), \tag{9.23}$$

and expanding the products:

$$\mathbf{I}\,\underline{\mathbf{x}}_R - \sum_{i=1}^{t}\left(\left(\mathbf{H_R}\,\mathbf{a_R^i}\,\mathbf{z^i} - \mathbf{H_I}\,\mathbf{a_I^i}\,\mathbf{z^i}\right)\underline{\mathbf{x}}_R - \left(\mathbf{H_R}\,\mathbf{a_I^i}\,\mathbf{z^i} + \mathbf{H_I}\,\mathbf{a_R^i}\,\mathbf{z^i}\right)\underline{\mathbf{x}}_I\right) + \\ + \left(\mathbf{I}\,\underline{\mathbf{x}}_I - \sum_{i=1}^{t}\left(\left(\mathbf{H_R}\,\mathbf{a_I^i}\,\mathbf{z^i} + \mathbf{H_I}\,\mathbf{a_R^i}\,\mathbf{z^i}\right)\underline{\mathbf{x}}_R + \left(\mathbf{H_R}\,\mathbf{a_R^i}\,\mathbf{z^i} - \mathbf{H_I}\,\mathbf{a_I^i}\,\mathbf{z^i}\right)\underline{\mathbf{x}}_I\right)\right) j = \\ = \underline{\mathbf{x}}_{0_R} + \underline{\mathbf{x}}_{0_I} j. \quad (9.24)$$

Notice that once the expression has been expanded, the identity matrix can be eliminated and this constraint can be split in two different equations, one equaling the real part of both sides and the other equaling the imaginary part, yielding:

$$\underline{\mathbf{x}}_R - \sum_{i=1}^{t}\left(\left(\mathbf{H_R}\,\mathbf{a_R^i}\,\mathbf{z^i} - \mathbf{H_I}\,\mathbf{a_I^i}\,\mathbf{z^i}\right)\underline{\mathbf{x}}_R - \left(\mathbf{H_R}\,\mathbf{a_I^i}\,\mathbf{z^i} + \mathbf{H_I}\,\mathbf{a_R^i}\,\mathbf{z^i}\right)\underline{\mathbf{x}}_I\right) = \underline{\mathbf{x}}_{0_R} \quad (9.25)$$

for the real part and

$$\underline{\mathbf{x}}_I - \sum_{i=1}^{t}\left(\left(\mathbf{H_R}\,\mathbf{a_I^i}\,\mathbf{z^i} + \mathbf{H_I}\,\mathbf{a_R^i}\,\mathbf{z^i}\right)\underline{\mathbf{x}}_R + \left(\mathbf{H_R}\,\mathbf{a_R^i}\,\mathbf{z^i} - \mathbf{H_I}\,\mathbf{a_I^i}\,\mathbf{z^i}\right)\underline{\mathbf{x}}_I\right) = \underline{\mathbf{x}}_{0_I} \quad (9.26)$$

for the imaginary part.

Now the equations are expressed just in real variables but they are still not linear constraints as there appear the products $\mathbf{z^i}\,\underline{\mathbf{x}}_R$ and $\mathbf{z^i}\,\underline{\mathbf{x}}_I$. Then, it will be desirable to deal with two new variables that represent these products. Let them be:

$$\underline{\mathbf{r}}^i = \mathbf{z^i}\,\underline{\mathbf{x}}_R \text{ and} \quad (9.27)$$

$$\underline{\mathbf{q}}^i = \mathbf{z^i}\,\underline{\mathbf{x}}_I. \quad (9.28)$$

Introducing them to both constraints (equation 9.25 and equation 9.26) they become linear equations:

$$\underline{\mathbf{x}}_R - \sum_{i=1}^{t}\left(\left(\mathbf{H_R}\,\mathbf{a_R^i} - \mathbf{H_I}\,\mathbf{a_I^i}\right)\underline{\mathbf{r}}^i - \left(\mathbf{H_R}\,\mathbf{a_I^i} + \mathbf{H_I}\,\mathbf{a_R^i}\right)\underline{\mathbf{q}}^i\right) = \underline{\mathbf{x}}_{0_R} \quad (9.29)$$

$$\underline{\mathbf{x}}_I - \sum_{i=1}^{t}\left(\left(\mathbf{H_R}\,\mathbf{a_I^i} + \mathbf{H_I}\,\mathbf{a_R^i}\right)\underline{\mathbf{r}}^i + \left(\mathbf{H_R}\,\mathbf{a_R^i} - \mathbf{H_I}\,\mathbf{a_I^i}\right)\underline{\mathbf{q}}^i\right) = \underline{\mathbf{x}}_{0_I} \quad (9.30)$$

Obviously, these new variables cannot be defined directly as the product presented in equation 9.27 and equation 9.28, because the linearity will be broken again.

Then, so as to achieve this equality just by linear relationships, some special properties of these products have to be taken into account:

- First, one of the variables in the product is a binary variable ($\mathbf{z^i}$).
- Second, the other variable ($\underline{\mathbf{x}}_R$ or $\underline{\mathbf{x}}_I$) is a real variable which can be bounded.

With respect to this second item, some explanation should be done:

As the objective is to reduce the vibration of the so called objective points (those belonging to the set $\mathcal{O}$), we can say that the modulus of each component of the complex vector $\underline{\mathbf{x}}$ for a feasible configuration must be smaller than the greater modulus of the components of the initial vibration vector ($\underline{\mathbf{x}}_0$). Let B be this bound:

$$B = \max\{\|x_{0_j}\|\} \quad for \quad j \in \mathcal{O}. \tag{9.31}$$

Then, as there has been explained above

$$\|x_j\| \leq B \quad for \quad j \in \mathcal{O}, \tag{9.32}$$

and as is true that

$$\|x_{R_j} + x_{I_j} j\| = \|x_j\| \Longrightarrow \begin{cases} |\ x_{R_j}\ | \leq \|x_j\| \\ |\ x_{I_j}\ | \leq \|x_j\|, \end{cases} \tag{9.33}$$

it can be said that:

$$\|x_{R_j} + x_{I_j} j\| = \|x_j\| \leq B \Longrightarrow \begin{cases} |\ x_{R_j}\ | \leq B \\ |\ x_{I_j}\ | \leq B \end{cases}, \tag{9.34}$$

or without the need of the absolute value operation

$$-B \leq x_{R_j} \leq B \tag{9.35}$$

$$-B \leq x_{I_j} \leq B. \tag{9.36}$$

On the other hand, the points belonging to the set $\mathcal{D}$ don't need to reduced their vibration although it could be desirable in some problems.

Then, so as to bound these values in a similar way as it has been done above with the points belonging to the set $\mathcal{O}$, three different options are presented:

a) Bounding the vibration of these points with the maximum initial vibration modulus of all the points in the study. Let this bound be B_a,

$$B_a = \max\{\|x_{0_j}\|\} \quad for \quad j \in \mathcal{O} \cup \mathcal{D}. \tag{9.37}$$

b) Giving these points some margin to increase the modulus of the vibration until a certain multiple of the bound B_a presented in equation 9.37,

$$B_b = n\ B_a. \tag{9.38}$$

c) Introducing a new bound to this set of points based in the experience of the engineer or in an other special criteria for a the particular case. Let it be B_c.

The choice of one of these options highly depends on the case in study and the analysis of them is beyond the scope of this project.

For a general case formulation let it be B_d the bound chosen from the three presented above.

$$B_d \in \{B_a, B_b, B_c\} \tag{9.39}$$

Once the bounds of the $\underline{\mathbf{x}}$ variable is settled and is known that $\mathbf{z^i}$ is a matrix of binary variables, it is possible to define the variables presented in equation 9.27 and equation 9.28.

On the one hand, as it is not possible to place a device in a point belonging to the set of objective points ($\mathcal{O}$), the binary variables belonging to these points will be null:

$$z_j^i = 0\ for\ j \in \mathcal{O},\ i \in \mathcal{T}. \tag{9.40}$$

Then, the variables r_j^i and q_j^i belonging to the set $\mathcal{O}$ will be always null:

$$r_j^i = 0\ for\ j \in \mathcal{O},\ i \in \mathcal{T} \tag{9.41}$$

$$q_j^i = 0\ for\ j \in \mathcal{O},\ i \in \mathcal{T}. \tag{9.42}$$

On the other hand, the components of the variables $\underline{\mathbf{r}}^i$ and $\underline{\mathbf{q}}_j^i$ belonging to the set of points $\mathcal{D}$ can be defined with a system of four linear inequalities:

$$r_j^i \leq B_d\ z_j^i\ for\ j \in \mathcal{O},\ i \in \mathcal{T}, \tag{9.43}$$

$$r_j^i \geq -B_d\ z_j^i\ for\ j \in \mathcal{O},\ i \in \mathcal{T}, \tag{9.44}$$

$$r_j^i \leq x_{R_j} + B_d(1 - z_j^i)\ for\ j \in \mathcal{O},\ i \in \mathcal{T}, \tag{9.45}$$

$$r_j^i \geq x_{R_j} - B_d(1 - z_j^i)\ for\ j \in \mathcal{O},\ i \in \mathcal{T}, \tag{9.46}$$

and

$$q_j^i \leq B_d \, z_j^i \; for \; j \in \mathcal{O}, \; i \in \mathcal{T}, \tag{9.47}$$

$$q_j^i \geq -B_d \, z_j^i \; for \; j \in \mathcal{O}, \; i \in \mathcal{T}, \tag{9.48}$$

$$q_j^i \leq x_{R_j} + B_d(1 - z_j^i) \; for \; j \in \mathcal{O}, \; i \in \mathcal{T}, \tag{9.49}$$

$$q_j^i \geq x_{R_j} - B_d(1 - z_j^i) \; for \; j \in \mathcal{O}, \; i \in \mathcal{T}. \tag{9.50}$$

It is easy to prove that with these new constraints the variables $\underline{\mathbf{r}}^i$ and $\underline{\mathbf{q}}^i$ achieve the desired value for the points belonging to $\mathcal{D}$:

if $z_j^i = 0$, from equation 9.43 and equation 9.44:

$$0 \leq r_j^i \leq 0 \Longrightarrow r_j^i = 0,$$

and this fulfills the constraints in equation 9.45 and equation 9.46:

$$x_{R_j} - B_d(1 - z_j^i) \leq r_j^i \leq x_{R_j} + B_d(1 - z_j^i).$$

And if $z_j^i = 1$, from equation 9.45 and equation 9.46:

$$x_{R_j} \leq r_j^i \leq x_{R_j} \Longrightarrow r_j^i = x_{R_j},$$

which also fulfills the constraints in equation 9.43 and equation 9.44:

$$-B_d \leq r_j^i \leq B_d,$$

and the same reasoning can be done with the component of $\underline{\mathbf{q}}^i$.

9.2 Implementation of the genetic algorithm

The genetic algorithm is implemented entirely in Matlab. It is built as a generic group of independent functions in order to achieve an optimization process that can be adapted to any of the problems that are laid out. Moreover, some functions, as the basic framework of the genetic algorithm (section 9.2.2), are build in a way that can be reused to totally different problems.

The whole process can be divided in two different parts in order to understand better how it works:

1. Data reading, previous computations and setting up.
2. Genetic algorithm loop.

The first part reads the data of the problem, makes the previous calculations required and sets up the parameters needed by the algorithm to operate (section 9.2.1). The second one calls the genetic algorithm function which performs the main iteration loop of the optimization process (section 9.2.2) by properly calling the functions that execute the different strategies (section 9.2.3) and the auxiliary functions that manage the data that will get the user (section 9.2.4).

9.2.1 Data reading, previous computations and setting up

This first part of the process starts by defining the constants of the problem and the parameters controlling the algorithm operation. All them are explained below.

Problem constants:

`types`: total amount of different types of absorber devices that will be in the problem (including rigid masses). It corresponds with the size of the set $\mathcal{T}$ defined in equation 9.3.

`ndmax`: maximum number of devices that can be placed in the structure. It corresponds with $n_{d_{max}}$ in equation 9.4.

`positions`: amount of different positions in which a device can be placed. Corresponds with m_p, the size of the set $\mathcal{D}$ defined in equation 9.2.

`op`: number of objective points in the problem. Corresponds with m_o, the size of the set $\mathcal{O}$ defined in equation 9.1.

`w`: working frequency in Hertz.

Algorithm parameters:

`pd`: number in the unit interval which selects the moment when the diversity criteria is off in the procreation strategy (see section 9.2.3.2).

`cramer`: boolean indicating if the fitness function will use Cramer's rule to solve the system or not (see section 9.2.5).

`maxg`: maximum number of iterations performed by the generic algorithm (see section 9.2.2).

`popsize`: population size used in the genetic algorithm (see section 9.2.2).

`en`: number of elite individuals in the genetic algorithm (see section 9.2.2).

`mr`: mutation ratio for the genetic algorithm (see section 9.2.2).

`pr`: procreation ratio for the genetic algorithm (see section 9.2.2).

`objective`**:** desired objective in optimization (see section 9.2.2).

`ninfo`**:** control of printing information frequency (see section 9.2.2).

`outputfile`**:** string variable containing the full name of the file where the output data will be saved (see section 9.2.2).

After these parameters are defined, the information needed by the algorithm is read from text files and the computations needed are performed:

- Input force vector: the force applied to the structure is read from the file *F.txt*.
- FRF matix: both real and imaginary parts of the FRF matrix are read from different files (*FRFMatrixR.txt* and *FRFMatrixI.txt*). And the complex FRF matrix ($\mathbf{H}$) is build with them.
- Devices physical properties: the physical properties needed to compute the equivalent mass (equation 6.2) (mass, spring stiffness and coefficient of viscous damping) are read for each different kind of device from the file *TMA.txt*. It must contain the number of different devices specified by the constant problem `types` introduced above.

For more information about the expected format of these files see annex C.

With the devices physical properties, the value of equivalent mass (equation 6.2) is computed for each different type. Then all the functions that will be needed are defined and those inputs that are constants are given to them before calling the genetic algorithm in order to save memory and increase the efficiency (as less parameters are passed between function as more faster the function call is done). Finally, a random initial population is generated using the function `randomind` (see section 9.2.3.4).

9.2.2 Genetic algorithm function

The genetic algorithm is build as a function in MATLAB. The objective is to make an algorithm as much generic as possible. In order to achieve that, this function requires a great amount of inputs. These inputs can be of different nature.

There are **control parameters** that control the performance of the population, the repopulation strategies or some operators features:

`popsize`**:** population size,

`mr`**:** mutation ratio,

`pr`**:** procreation ratio,

`en`: number of elite individuals,

`pd`: indicates the moment in which the diversity criteria will be disabled (see section 9.2.3.2).,

the **initial population**:

`pop`: contains the individuals to start the algorithm,

the **break parameters** that indicate when the algorithm has to stop:

`maxg`: maximum generations to compute,

`objective`: desired objective in optimization,

the **functions** to perform the different strategies

`computefitness`: function to compute the fitness of an individual,

`randomind`: function to generate a random but feasible individual,

`mutation`: function to create a new individual by the mutation strategy,

`procreation`: function to create a new individual by the procreation strategy,

`uniqueness`: function to check if all individuals in the population are unique,

and the **output parameters** that settle the desired output information:

`printdata`: function that defines the output data and the format with which it will bew printed,

`ninfo`: control of printing information frequency,

`outputfile`: string variable containing the full name of the file where the output data will be saved.

The process performed by the algorithm is schematized in figure 9.1. The first step is to compute the fitness of the initial population entered as an input and to sort this population by the fitness value of each individual. As this is a minimization algorithm, the population will be sorted from the lower value of fitness to the higher. At this time, the generations counter (`ng`) is initialized to 1 and it starts the main loop of the algorithm.

This loop started with four different checks. First, it is checked if the population is degenerated. If the population size is less than the size of the elite population, it is considered that the algorithm has degenerated and the loop will finish printing the data request by the `printdata` function together with the message *"STOP: Population degenerated"*. If the size of the population is correct, it is checked if the desired objective has been reached.

It is done by comparing the fitness value of the first individual with the value entered by the variable `objective`. If this fitness is lower or equal than the desired value, the loop is finished and it is printed the last data request by `printdata` and the message *"STOP: Objective reached"*. Otherwise, the algorithm will continue checking if the maximum number of iterations (`maxg`) has been performed. This check consists in comparing the generation number (`ng`) with the maximum number of iterations. If both numbers are equal, the algorithm finished printing the message *"STOP: Maximum iteration performed"* and the data of the last iteration with the `printdata` function. If the loop doesn't have to end, it is checked if it is requested to print data. Here it is tested if the iteration number (`ng`) is a multiple of the variable `ninfo`, which controls the frequency of printing information. The function `printdata` will be called only if it is true, therefore it will print information only every `ninfo` iterations. Notice that both, if the information is printed or not, the algorithm will continue the loop in this case.

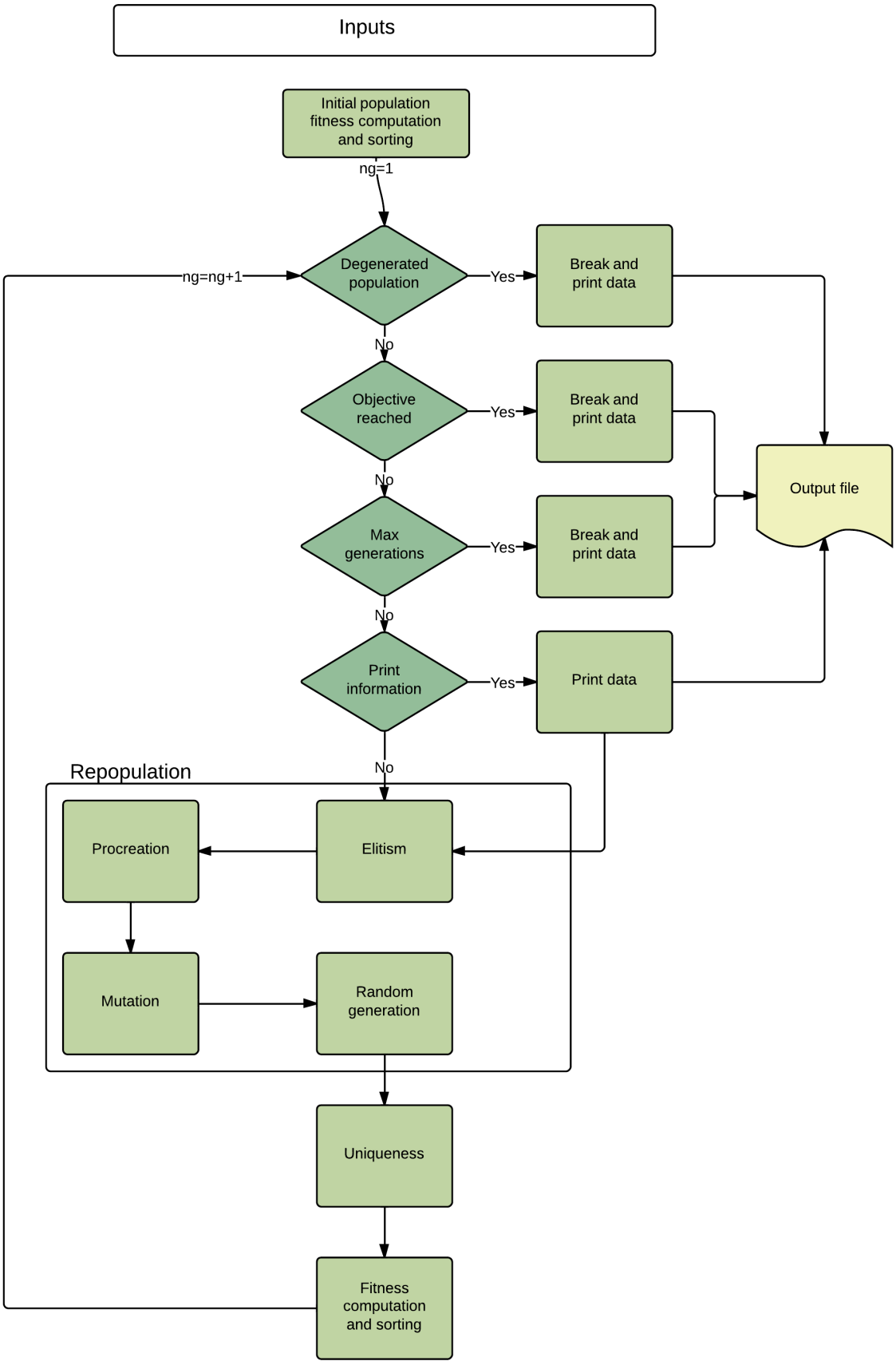

Figure 9.1: Genetic algorithm diagram.

Once all the checks are done, the next step is to create the next generation. It is done by a repoputation process that calls the different strategies following the next sequence:

1. Elitism: the first `en` individuals are directly copied to the new population.
2. Procreation: the procreation function is called to create new individuals. The number of individuals created by procreation is computed with the procreation ratio (`pr`). It is the result of rounding up the product `popsize pr`.
3. Mutation: the mutation function is called to create new individuals. In a similar way to the procreation, the number of new individuals generated by mutation is computed by rounding up the product between the population size (`popsize`) and the mutation ratio (`mr`).
4. Random generation: the remaining individuals, until the desired population size (`popsize`) is reached, are generated in a totally random way by calling the `randomind` function.

After generating a new population, the next step is to check if all individuals in it are unique. It will be done by calling the function `uniqueness`, which eliminate the repeated individuals from the population.

Once the population contains only unique individuals, the loop finishes computing the fitness of all the individuals in the new population and sorting them from the lower fitness to the higher. Then, the counter of the generations performed increases in one and the loop starts again with the check of degenerated population.

9.2.3 Strategies

9.2.3.1 Elitism

Elitism is a simple strategy that consists in coping the best individuals from the current population to the next one. The number of individuals copied is given by the parameter `en`.

This strategy is very useful in this kind of problems with integer variables where there is a very low probability of producing again the same individual and the fitness value could change a lot for individuals that are close in the searching space. The reason of this is that elitism ensures that the best fit found is never lost.

9.2.3.2 Procreation

The procreation strategy is programmed with two different operation modes. The first one takes into account the diversity and the fitness of the individuals to compute their probability of reproduction; the other one just attends to the fitness.

It has been also build as a MATLAB function with **inputs**:

`pop`**:** current individual population,

`fi`**:** array containing the fitness value of the corresponding individual in the same position in pop,

`pdg`**:** a parameter in the unit interval that indicates the state of the general algorithm as in the mutation function,

`pd`**:** value of pdg from which the diversity criteria will be off,

and returning as **output**:

`NewI`**:** newcomer individual generated.

The first step is computing the so called relative fitness. This parameter is valued between 0 and 1 and represents how good is an individual compared with the rest of the population. To compute it, as the fitness values are always positive numbers usually less than 1 and the objective is to minimize it, the inverse of each fitness value is taken and divided for the sum of the inverses of the whole population:

$$\texttt{rfi}_i = \frac{\frac{1}{\texttt{fi}_i}}{\sum_{j=1}^{n} \frac{1}{\texttt{fi}_j}} \quad for \;\; i = 1, ..., n, \tag{9.51}$$

where n is the population size.

Then, it is checked if the probability must be computed taking into account the diversity or just the fitness. If the value of`pdg` is less or equal than `pd`, the diversity will be taken into account, otherwise it won't.

The criteria to value the diversity of an individual with respect to the rest of the population is just the amount of devices placed. First, it is computed the average number of rows of the population individuals (ms), and then, the diversity is computed as

$$\texttt{div}_i = \frac{\mid \text{size}(\texttt{pop}_i) - ms \mid}{ms\; n} \quad for \;\; i = 1, ..., n, \tag{9.52}$$

where size$(\texttt{pop}_i)$ is the number of rows of the individual i and n is the population size as

in equation 9.51. Notice that using the absolute value operation and dividing the value by $(ms\ n)$ ensures that it will be in the unit interval.

The diversity is computed this way because is the only criteria that gives a good estimation of the diversity without being very expensive computationally speaking.

Once it is computed, the probability of reproduction of an individual is defined by the product of its relative fitness and its diversity,

$$\mathtt{p}_i = \frac{\mathtt{rfi}_i\ \mathtt{div}_i}{\sum_{j=1}^{n}(\mathtt{rfi}_j\ \mathtt{div}_j)} \quad for\ \ i = 1, ..., n. \tag{9.53}$$

On the other hand, if the value of `pdg` is greater than `pd`, the diversity criteria is off and the probability is directly the value of the relative fitness

$$\mathtt{p}_i = \mathtt{rfi}_i \quad for\ \ i = 1, ..., n \tag{9.54}$$

This process is outlined below in figure 9.2.

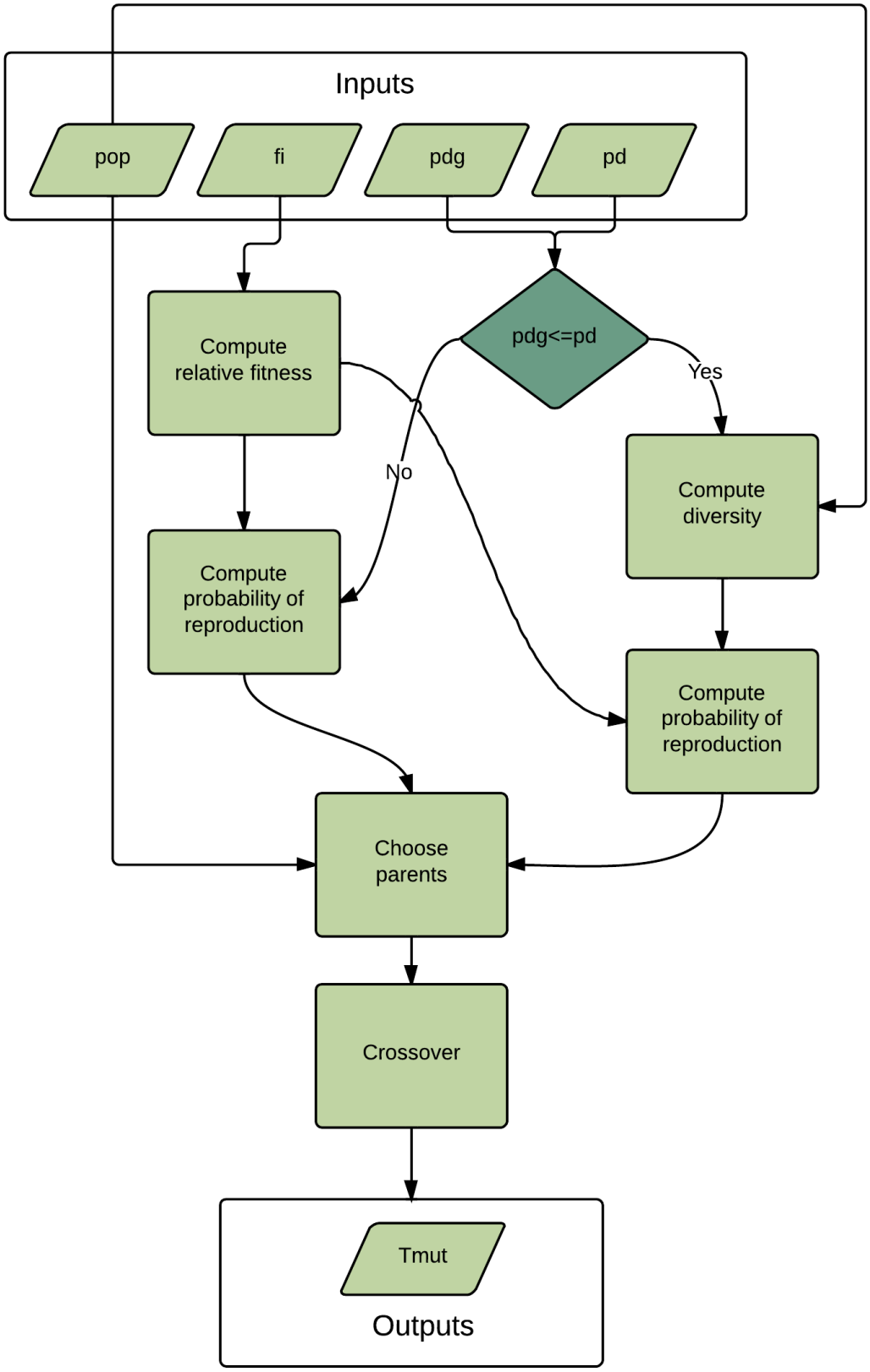

Figure 9.2: Procreation function diagram.

Once the probability is computed by one of the criteria explained above, two individuals from the population are chosen. Each individual have the probability of being chosen that has been computed. This process is implemented by the MATLAB function `randsample`(n,2,true,p) which returns a sample of two integers from 1 to n taken with replacements and with the probability of being selected given by the array p [17].

Finally, the chosen individuals are crossed over. This process has to be very careful in order to not generate a new individual with more than one device in the same position. Firstly, the fist half of the rows (computed by integer division) from the first individual chosen is copied to the newcomer. Then, for each row of the second individual chosen (starting for the first one) it is checked if there is any device already placed in that position; if there is not, the entire row is copied to the new individual. This loop ends both if all the rows of the second individual are checked or if the amount of rows copied from the second parent is greater than the half of its size (computed again by integer division). Once this process is finished, the new individual generated is returned.

9.2.3.3 Mutation

The mutation strategy is programmed as a MATLAB function. Its **inputs** are:

`pop`: the current population,

`types`: the number of different types (t),[1]

`positions`: the number of possible positions (m_p),[1]

`pdg`: a parameter between 0 and 1 that indicates the state of the general algorithm. It is defined as the ratio between the generation number and the maximum generations that could be performed:

$$\mathrm{pdg} = \frac{\mathrm{ng}}{\mathrm{maxg}},$$

and returns the **output**

`Tmut`: mutated individual.

The process followed by this function to create the new individual is outlined in figure 9.3. The first steps are choosing the individual to mutate and sizing the mutation. The individual to mutate is randomly chosen from the current population. The size of the mutation is decreasing along the algorithm process in order to generate more diversity between individuals during the first iterations and to reduce it eventually. This is a concept taken from simulated annealing algorithms.

[1] These parameters are included so as to make the algorithm as much generic as possible.

So as to do this, it has been built the exponential law

$$k = 1 + \exp(\texttt{pdg c}), \tag{9.55}$$

where c is a parameter to determine the speed of this reduction. The size (number of rows) of the chosen individual is divided by this value (k) and the result of this operation rounded to the nearest greater or equal integer gives the number of rows that will be changed.

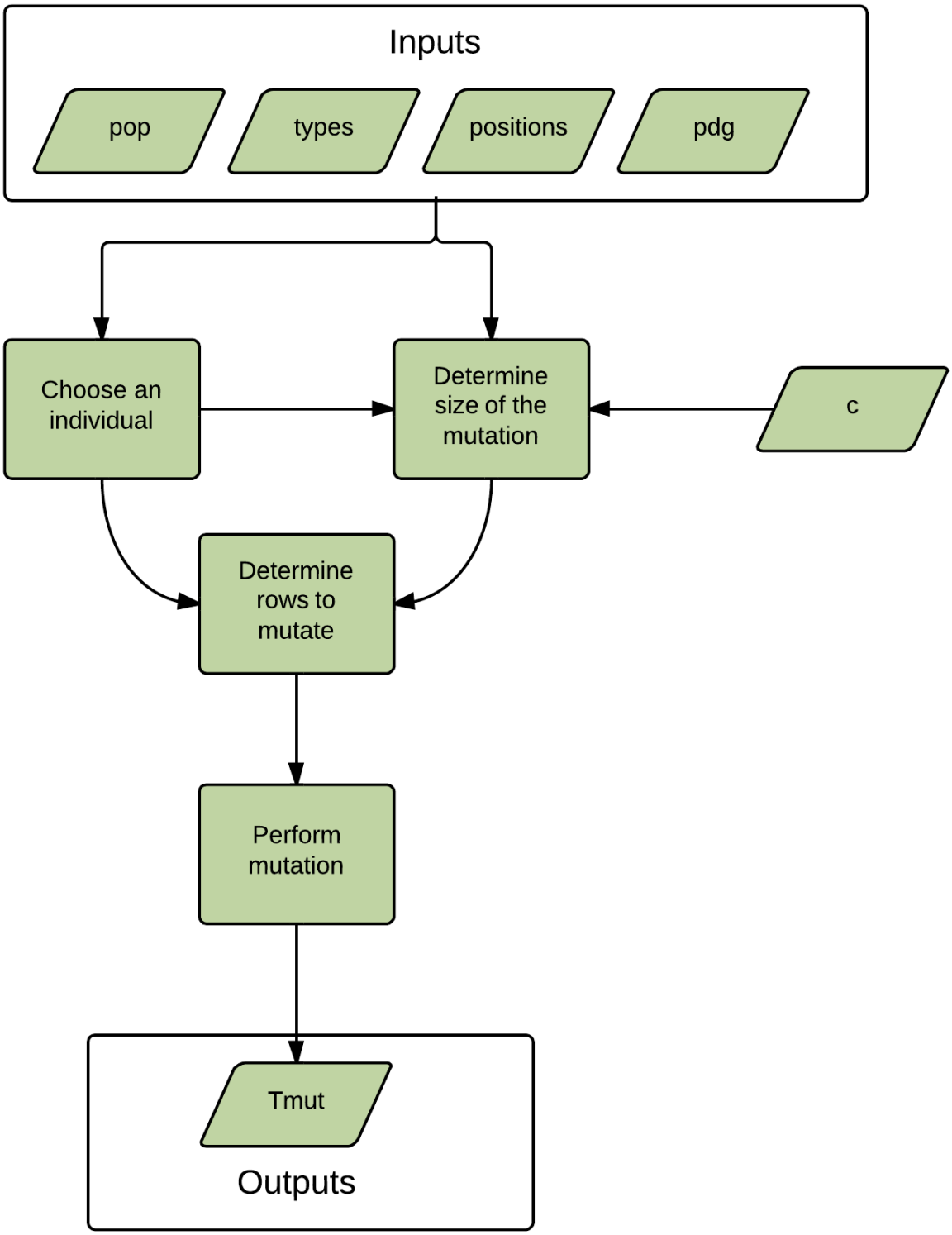

Figure 9.3: Mutation function diagram.

Once the size of the mutation is known, it is randomly selected as many integers as number of rows that will be changed (between 1 and the size of the individual). Notice

that this process is programmed in such a way that it is possible to change more than once the same row in the same mutation. Finally, the type of the chosen rows is randomly settled giving the mutated individual (`Tmut`).

9.2.3.4 Random generation

The random generation is built as a MATLAB function. It generates a completely random individual with the desired format which is contained in the search space.

The **inputs** of this function are:

`ndmax`**:** maximum number of devices that can be placed in the structure ($n_{d_{max}}$),

`types`**:** the number of different types (t),

`positions`**:** the number of possible positions (m_p),

and returns as **output**:

`randT`**:** random individual generated.

First, the size of the new individual (n_i) is randomly determined as an integer between 1 and `ndmax`. After this, it is generated a random array of size n_i containing the integers which defines the types of the devices, they will be integers between 1 and `types`. Then, the positions in which that devices will be placed are defined by an random permutation array of integers from 1 to `positions` so as to not repeat any position.

9.2.3.5 Uniqueness function

The uniqueness function has also been built as a MATLAB functions. The function uses all the power of MATLAB to eliminate repeated individuals in the population.

To do this, every individual in the population is sorted by position in ascending order. Once all the individuals are sorted, as there cannot be more than one device placed in the same position, it can be checked if two individuals are the same just comparing if both matrices are exactly equal (what can be done very efficiently in MATLAB)[2].

The uniqueness function saves a lot of time in those kind of problems where a huge time is needed to compute the fitness of an individual compared with the time expended in calling this function. It also allows the algorithm to stop if the population is degenerated.

[2] See `isequal` function in [17].

9.2.4 Auxiliary functions

9.2.4.1 Printdata function

The `printdata` function is a MATLAB function that sets the format of the printed information in a given output file.

The inputs required are:

`ng`: generations counter,

`OFile`: variable containing the full information about the desired output file already opened and ready to write,

`individual`: individual that is going to be printed. Usually it corresponds with the best individual in the population but it could be easily changed in the algorithm function (section 9.2.2),

`fitness`: variable containing the fitness value of the individual printed. As the individual is usually the best individual in the population and the algorithm in use is elitist, this fitness is usually the best fitness achieved until the moment.

The performance of this function consist in writing in the `OFile` a first line with the iteration number given by `ng` and the fitness value given by `fitness`. In the second line the elapsed time between the start of the algorithm and the current moment is printed. Finally, the whole `individual` matrix is printed as can be seen in figure 9.4

```
Iteration 1000.  Fitness: 3.08396838e-05
Time: 92.8794 sec
Best individual:
2   1
2   2
2   3
2   7
3   9
3   10
3   11
2   12
2   16
2   19
2   20
2   24
2   25
3   28
2   29
2   30
2   32
-----------------------------------
```

Figure 9.4: Sample of the data format generated by the `printdata` function.

9.2.5 Fitness function

The fitness function is build as a MATLAB function. This function solves the system from equation 9.9 for a given individual and computes its fitness value as defined in equation 9.8.

The **inputs** required by this function are;

`T`: individual to be evaluated,

`meq`: array containing the equivalent masses of all the possible types of device,

`H`: FRF matrix,

`positions`: amount of different positions in which a device can be placed,

`op`: number of objective points in the problem,

`F`: input force vector,

`w`: working frequency in Hertz,

`cramer`: boolean indicating if the system will be solved by Cramer's rule or not,

and it returns as **output**:

`fitness`: value of the objective function computed by equation 6.21 for the given individual.

The first step performed by this function is decoding the individual. Then, the system is build: matrix $\mathbf{a}$ is created by assembling the corresponding values of m_{eq} in the corresponding position of the matrix and also the right hand side vector ($\underline{\mathbf{x}}_0$) is computed by equation 6.5.

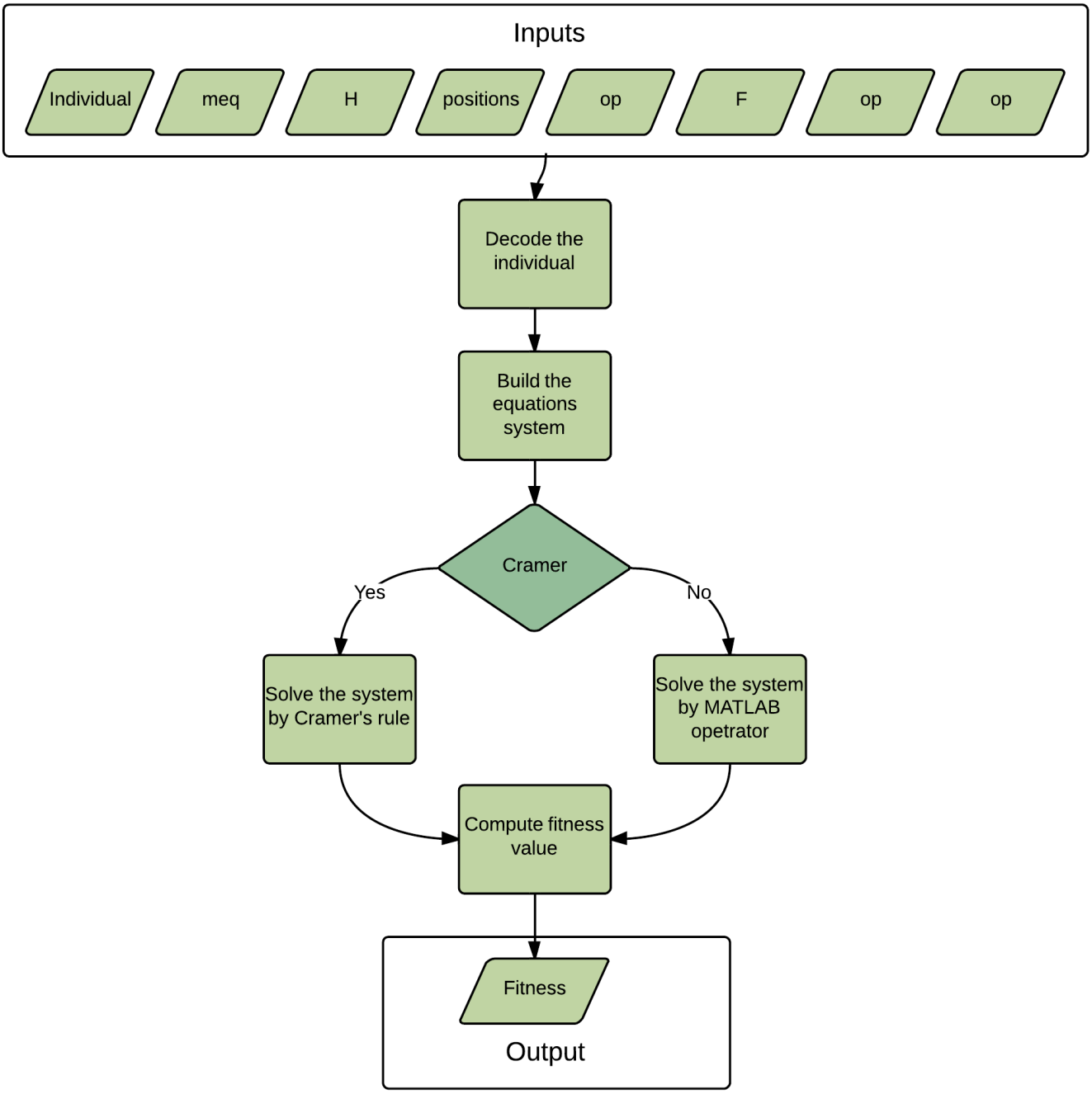

Figure 9.5: Fitness function diagram

After this, it is checked the boolean `cramer`. If it is activated, the system will be solved by Cramer's rule only for the objective points; otherwise, the system will be fully solve by the MATLAB command `mldivide`, especially build to solve linear systems (see [17]).

Finally, the fitness value is computed as indicated in equation 9.8 and returned to the genetic algorithm function (see section 9.2.2). All this process is outlined in figure 9.5.

9.2.6 Validation of the genetic algorithm

In order to validate the genetic algorithm that has been implemented a simple case for which is feasible to compute all the possible combinations has been optimized.

The case is the same used in section 6.4; a plane plate where there are 9 points in where there can be devices and there is only one type of device available. In this section the data of the problem is not very relevant as the aim is to prove the well operation of the algorithm, therefore the physical properties of the plate and the DVA are presented in annex Band there are not discussed the parameters of the algorithm.

The best fitness value found is $8.1706 \cdot 10^{-5}$ m^2 for the individual

$$\begin{matrix} 1 & 7 \\ 1 & 8 \end{matrix},$$

which agrees with the minimum found in section 6.4.

Therefore, it is proved that the genetic algorithm implemented works properly and is able to found the global minimum of the function. However, due to the simplicity of the problem solved, there cannot be anything conclude about the efficiency or the benefits of using the algorithm from the results.

9.3 Implementation of the problem in CPLEX

The MIQP problem is implemented in the CPLEX Python API. The application has been programmed in a generic way so as to allow this program to solve any problem if the different inputs needed and the different parameters are properly setted up. In this section it is explained the structure of the application as well as its operation.

In order to explain it properly, it can be internally divided in three differentiated parts:

1. Data reading, previous computations and setting up.
2. Definition of the problem.
3. Solution and results processing.

Each of them are explained in detail in the next sections.

9.3.1 Data reading, previous computations and setting up

In this first part of the application all the parameters that define the particular problem are settled , all the information needed to the formulation of the problem is read and all the previous computations required are performed.

The parameters that have to be defined for a correct operation of the application are:

- `mo`: number of objective points (size of the set $\mathcal{O}$ defined in equation 9.1),
- `mp`: number of absorber points (size of the set $\mathcal{D}$ defined in equation 9.2),
- `t`: number of different absorber types available (size of the set $\mathcal{T}$ defined in equation 9.3),
- `nd_max`: maximum number of devices that can be placed in the structure,
- `w`: working frequency (in Hertz)

Once this parameters are known the application reads the information that is needed to formulate the problem, it is:

- FRF matrixes: the real and imaginary parts of the frequency response functions are read separately from de files *FRFMatrixR.txt* and *FRFMatrixI.txt*.
- Devices physical characteristics: the physical characteristics needed to compute the equivalent mass (equation 6.2) (mass, spring stiffness and coefficient of viscous damping) are read for each different kind of device from the file *TMA.txt*. It must contain the number of different devices specified by the parameter `t` introduced above.
- Input force vector: the force applied to the structure is read from the file *F.txt*. Its dimensions must agree with the FRF matrix.

For more information about the expected format of these files see annex C.

Concurrently it is compute:

- $\underline{\mathbf{x}}_{0_R}$ and $\underline{\mathbf{x}}_{0_I}$ vectors following equation 6.5,
- the equivalent masses $m_{eq}^i \ for \ i \in \mathcal{T}$ that constitute the matrices $\mathbf{a^i} \ for \ i \in \mathcal{T}$ by equation 6.2 and
- the bounds of the variables related to $\underline{\mathbf{x}}$ vector (B and B_d defined in equations equation 9.31 and equation 9.39).

9.3.2 Definition of the problem

Once all the data is available and the previous computations are done, the application proceeds to define the MIQP problem in CPLEX.

The first step is to properly set up the parameters that will be used to solve the problem. CPLEX has a great amount of parameters that can be changed but discussing them is out of the scope of this project (for further information about them see [18]). For a proper set up of the problem, the only parameters that are changed from their default value are some of the related with the tolerance:

- Relative MIP gap tolerance (`mipgap`): this parameter sets the moment in which a non integer solution is considered close enough from an integer solution so as to stop the problem in relative terms (expressed as per-unit). It is set on zero because a pure integer solution is desired and little variations from them can give a false optimal as the value of the objective function is usually a very low (around 10^{-6} or 10^{-8}) and little perturbations can highly distort it.

- Absolute MIP gap tolerance(`absmipgap`):this parameter sets the moment in which a non integer solution is considered close enough from an integer solution so as to stop the problem like the `mipgap` parameter but in absolute terms. It is also set on zero so as to obtain a pure integer solution as explained above.

- Feasibility tolerance (`feasibility`): this parameter specifies how the value of a basic variable calculated by the method may violate their bounds. It is set on the minimum value possible (10^{-9}). The reason of setting it to its minimum is that some of the variables the problem is dealing with have a low value (around 10^{-6} or 10^{-8}). Then, this new value prevents errors on the equations that may make the algorithm find a non real optimum solution. However, if the application reports problems of unfeasibility for a particular case they could be solved by readjusting this value.

- Integrality tolerance (`integrality`): this parameter specifies the amount by which a integer variable can be different from an integer and still be considered feasible. This tolerance is set on zero because as explained with `mipgap`, some little perturbations in the variables can have a hight effect on the overall problem. For example, if the binary variables that have to represent that a device is not placed is not completely zero (in machine precision), it will slightly affect the objective function value; but if all the devices which binary variable (z_j^i) must be zero have a little contribution, it will highly affect the obtained result giving a false optimal solution.

After setting up the parameters for the problem the constraints are populated. This process has been build in three different blocks.

In the first block the dynamic relationships for both the real part (equation 9.29) and the imaginary part (equation 9.30) are populated. The first step is to define the variables conforming the vectors related with the displacements ($\underline{\mathbf{x}}$). In particular:

- x_{R_j} is defined for each point j in the set $\mathcal{O} \cup \mathcal{D}$.
- x_{I_j} is defined for each point j in the set $\mathcal{O} \cup \mathcal{D}$.
- r_j^i is defined for each type i in $\mathcal{T}$ and for each point j in $\mathcal{D}$. Notice that the variable belonging to the objective points set ($\mathcal{O}$) are not defined as they are always null.
- q_j^i is defined for each type i in $\mathcal{T}$ and for each point j in $\mathcal{D}$. The ones belonging to the set $\mathcal{O}$ are not defined for the same reason explained above with r_j^i.

All these variables are defined in the application with the corresponding upper and lower bounds, given by equation 9.31 and equation 9.39.

Linear constraints have to be defined in CPLEX as a simple matrix system expression like

$$\mathbf{A}\ \underline{\mathbf{x}} \sim \underline{\mathbf{b}}$$

where $\mathbf{A}$ is a matrix containing the constraint coefficients, $\underline{\mathbf{x}}$ represent the variables vector, $\underline{\mathbf{b}}$ the right hand sides vector and the symbol $\sim$ can be $\geq$,$\leq$ or $=$, which is represented by the sense of the constraint [19].

Then, as the relationships presented in equation 9.29 and equation 9.30 have not this shape, it is necessary to expand the matrix products in a generic way until a simple expression which defines a row of the system is obtained. To achieve that, it is expanded a simple case and the results are extrapolated to the whole system. The intermediate steps are omitted because they are tedious and don't contribute to the understanding of the problem. After this process, the next expression is found to represent the real part of the system for each row (j):

$$x_{R_j} + \sum_{i=1}^{t} \sum_{k=mo+1}^{mo+mp} \left[-\left(H_{R_{jk}}\, a_R^i - H_{I_{jk}}\, a_I^i\right)\ r_k^i + \left(H_{I_{jk}}\, a_R^i + H_{R_{jk}}\, a_I^i\right) q_k^i \right] = x_{0_{R_j}}, \quad (9.56)$$

and the next one represents the imaginary part also for each row (j):

$$x_{I_j} + \sum_{i=1}^{t} \sum_{k=mo+1}^{mo+mp} \left[-\left(H_{I_{jk}}\, a_R^i + H_{R_{jk}}\, a_I^i\right)\ r_k^i - \left(H_{R_{jk}}\, a_R^i - H_{I_{jk}}\, a_I^i\right) q_k^i \right] = x_{0_{I_j}}. \quad (9.57)$$

In these equations the coefficients taking each variable are easily identified and the expression can be build in the API by nested for loops.

In the second block the linear equations system that defines the relationship between the variables r_j^i and q_j^i with the displacement (x_{R_j} or x_{I_j} as appropriate) and the binary variable z_j^i for each point j in $\mathcal{D}$ and every type i in $\mathcal{T}$ given by equations 9.43 to 9.50 are formulated.

As it has been done with the continuous variables in the previous block, it is necessary to define the variables z_j^i as binary variables. Then, to define these constraints into the CPLEX Python API it is necessary to reformulate them in order to achieve the desired structure:

$$r_j^i - B_d\, z_j^i \leq 0 \; for \; j \in \mathcal{O},\; i \in \mathcal{T}, \tag{9.58}$$

$$r_j^i + B_d\, z_j^i \geq 0 \; for \; j \in \mathcal{O},\; i \in \mathcal{T}, \tag{9.59}$$

$$r_j^i - x_{R_j} + B_d\, z_j^i \leq B_d \; for \; j \in \mathcal{O},\; i \in \mathcal{T}, \tag{9.60}$$

$$r_j^i - x_{R_j} - B_d\, z_j^i \geq -B_d \; for \; j \in \mathcal{O},\; i \in \mathcal{T}, \tag{9.61}$$

and

$$q_j^i - B_d\, z_j^i \leq 0 \; for \; j \in \mathcal{O},\; i \in \mathcal{T}, \tag{9.62}$$

$$q_j^i + B_d\, z_j^i \geq 0 \; for \; j \in \mathcal{O},\; i \in \mathcal{T}, \tag{9.63}$$

$$q_j^i - x_{R_j} + B_d\, z_j^i \leq B_d \; for \; j \in \mathcal{O},\; i \in \mathcal{T}, \tag{9.64}$$

$$q_j^i - x_{R_j} - B_d\, z_j^i \geq -B_d \; for \; j \in \mathcal{O},\; i \in \mathcal{T}. \tag{9.65}$$

In the third and last block the remaining constraints defining the problem are populated, they are:

- The constraint introduced in equation 9.16, which represents that there only can be one device in a position for each positions in $\mathcal{D}$.
- The constraint introduced in equation 9.18, which sets the maximum devices that can be placed in the structure to n_{d_max}.

Finally, once all the constraints and variables are defined, the objective function and its sense are set. The sense of the objective is to minimize and the objective is the quadratic function Ψ represented by the equation 9.13 in the MIQP formulation or using the notation of the variables defined in the application (separating real and imaginary part in different variables),

$$\sum_{j=1}^{mo} \left(x_{R_j}^2 + x_{I_j}^2 \right). \tag{9.66}$$

9.3.3 Solution and results processing

In this last part of the application CPLEX is called to solve the problem and then the solution and some status indicators are printed on the screen. It is possible to call a great amount of information about the solution of the problem but this text is focusing in a few that are considered basic and indispensable. For further information consult [18]. The data presented by this application is:

- Solution status: numeric code and and string that gives informations about the reason for which the solver has stopped as well as the type of solution reached. For more information see [18].
- Iterations counter: number of iterations performed by the solver.
- Objective reached: Value of the objective function for the solution found.
- Solution: solution value of all the binary variables defining the device combination placed in the structure.

9.3.4 Validation of the model

In order to validate the model built it has been optimized a simple case where it is feasible to compute all the possible combinations.

The case is the same used in section 6.4; a plane plate where there are 9 points in where there can be devices and there is only one type of device available. In this section the data of the problem is not very relevant as the aim is to prove the well operation of the algorithm, therefore the physical properties of the plate and the DVA are presented in annex Band there are not discussed the parameters of the algorithm.

The optimal objective value found is $8.1706 \cdot 10^{-5}$ and the non zero binary variables are:

$$z_7^1 \quad z_8^1 \ ,$$

which mean that there are two devices placed in the plate; one in the node number 7 and the other in the number 8.

This results fully agree with the minimum found in section 6.4 where all the possibilities were computed. Therefore, it is proved that the model formulated in section 9.1.2 is correct and the implementation in the CPLEX Python API works properly.

However, due to the simplicity of the problem, the results are not relevant to make any conclusion about the efficiency of the application or its benefits. Then, they will not be discussed in this text.

9.4 Case study

In this section it is solved an optimization problem concerning a plane plate and two different types of DVA at a frequency of 80Hz. This case is an example which finality is to illustrate the operation of the optimization tools developed.

The plate material properties and dimensions are summarized in table 9.1, and figure 9.6 illustrates the geometry and the mesh of the structure. In table 9.2 there are the

Lx	Ly	Thickness	Material	Damp. coef.	ρ	ν	E
5m	3m	0.01m	Al2024	0.01	2780 $\frac{\text{kg}}{\text{m}^3}$	0.33	73.1$\cdot$ 10^9 Pa

Table 9.1: Physical properties of the plane plate studied.

physical properties defining the available DVAs for this optimization. The data is given in International System of Units, so mass is given in [kg], spring stiffness is given in $[\frac{\text{N}}{\text{m}}]$ and coefficient of viscous damping is given in $[\frac{\text{N s}}{\text{m}}]$. Type number 2 has been chosen to be tune at the working frequency and type number 1 is a similar device with 0.100kg less of mass.

Type	m[kg]	$k[\frac{\text{N}}{\text{m}}]$	$c[\frac{\text{N s}}{\text{m}}]$	Tuning ω[Hz]
1	0.150	63165.47	9.4876	103.3
2	0.250	63165.47	9.4876	80

Table 9.2: Physical properties of the available DVAs.

The boundary conditions applied to the model are restrictions in all three degrees of freedom of displacement in the perimeter nodes of the plate.

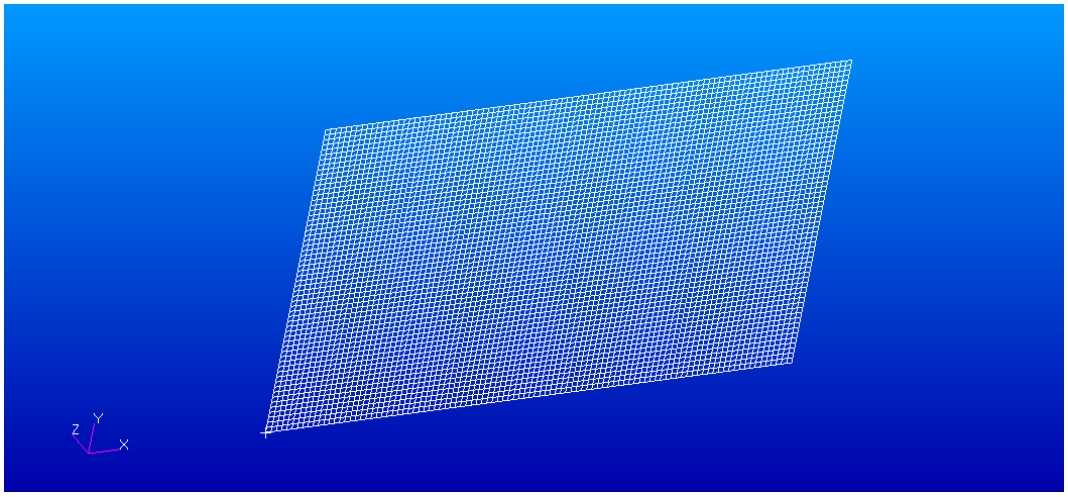

Figure 9.6: Case Geometry and Mesh

The main constants in the optimization problem are

- A total of thirty possible points where a device can be placed (m_p =30).
- A total of twenty-one objective points (m_o =21).
- A maximum amount of ten devices can be placed ($n_{d_{max}}$ =10).

The points of possible DVA allocation have been chosen in order to cover the whole surface of the plate, whereas the objective points, in which the vibration is reduced, are chosen to be representative of the lower right corner of the plate. The origin of coordinates has been placed at the lower left corner of the plate. This idea is illustrated in figure 9.7.

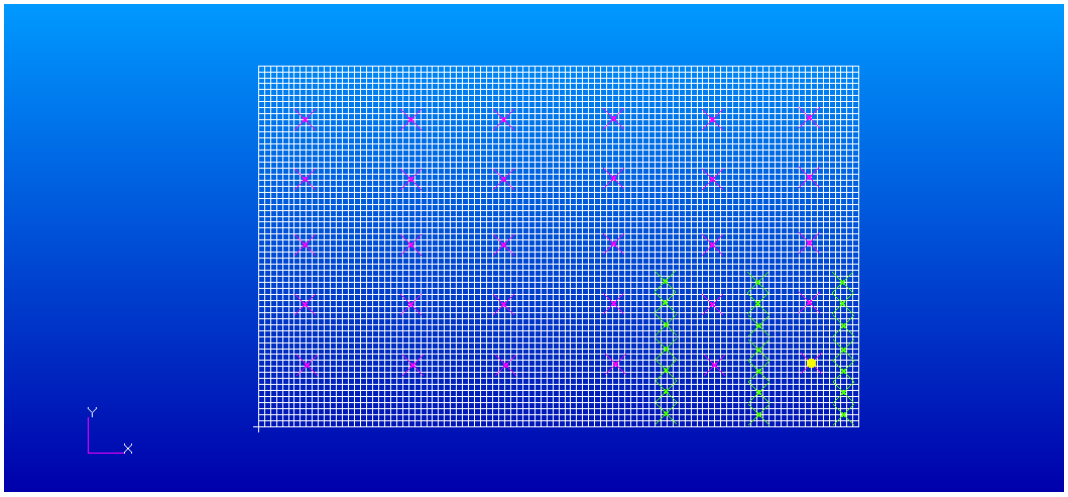

Figure 9.7: Significant points in the plate: objective points in green, possible points in red and the point where the force is applied in yellow.

The external force has been applied to a single node of the lower right corner of the plate. The force has a value of $250 \cdot 10^6$N and a frequency of 80Hz.

9.4.1 Preprocessor

Once the points of interest are known its closest node is chosen in the FEM model. Then, the Frequency Response Function (FRF) is computed for each node respect to all the others in order to conform the FRF Matrix.

This simulations are done with the NASTRAN software. To automatize this procedure it has been developed a Python script which creates the input file required by NASTRAN, calls the solver and treats the results so as to obtain the FRF matrix in the desired format. For a detailed explanation of this procedure see [2].

9.4.2 Optimization by the genetic algorithm

In order to solve this problem with the implemented GA, it is need to create the inputs files containing:

- the real part of the FRF matrix,
- the imaginary part of the FRF matrix,
- the input force vector,
- the physical properties of the available DVAs,

and this files must follow the format defined in annex C.

Once the problem is defined, the constants defining it have to be given to the algorithm. It is done by mean assigning the correspondent value to the variables of the program as showed in table 9.3.

types	2
ndmax	10
positions	30
op	21
w	80

Table 9.3: GA constants of the problem.

Then, the parameters of the algorithm have to be set up. As the aim of this case is to illustrate the proper operation of the algorithm but not to optimize its performance, the parameters controlling the strategy used are presented in order to allow anyone to repeat the test, but their discussion is out of the scope of this study. In table 9.4 the whole configuration is presented.

pd	0.5
maxg	500000
popsize	50
en	3
mr	0.3
pr	0.5
objective	0
cramer	0

Table 9.4: GA parameters set up for the case.

After performing the 500000 iterations in 9413.46 s, the best individual achieved is presented in table 9.5. Which has a fitness value of 1396.6 m^2.

This solution is equivalent to an insertion loss of

$$IL_{GA} = 6.22 \text{dB},$$

and it supposes an increase of weight of 2.3 kg.

Type	Node ID	XY Coordinates [m]
2	1	(0.5 , 2.5)
2	2	(0.5 , 2)
2	3	(0.5 , 1.5)
1	4	(0.5 , 1)
1	5	(0.5 , 0.5)
2	7	(1.5 , 2)
2	8	(1.5 , 1.5)
2	18	(3.5 , 1.5)
2	24	(4.5 , 1)
2	25	(4.5 , 0.5)

Table 9.5: Best individual achieved with the GA.

9.4.3 Optimization by MIQP

The optimization tool developed to call the CPLEX solver needs the same input files that the GA (FRF matrix, input force and DVA properties).

Then, the main variables of the project have to be given to the application (see section 9.3). They are presented in table 9.6.

`mo`	21
`mp`	30
`t`	2
`nd_max`	10
`w`	80

Table 9.6: Variables of problem introduced to MIQP application.

The solution achieved is the optimal solution and is presented in table 9.7. It has been codified as a GA individual in order to compare both solutions.

The objective value reached is 1230.1902 m^2, which represents an insertion loss of

$$\text{IL}_{MIQP} = 6.46\text{dB}.$$

It has been achieved in 4277.37s with $1.86 \cdot 10^6$ iterations. And this solution supposes an increment of 2.5 kg.

Type	Node ID	XY Coordinates [m]
2	1	(0.5 , 2.5)
2	2	(0.5 , 2)
2	7	(1.5 , 2)
2	8	(1.5 , 1.5)
2	15	(2.5 , 0.5)
2	18	(3.5 , 1.5)
2	19	(3.5 , 1)
2	22	(4.5 , 2)
2	24	(4.5 , 1)
2	25	(4.5 , 0.5)

Table 9.7: Optimal solution achieved by MIQP.

9.4.4 Analysis of results

The insertion loss achieved for both of the optimization tools shows that the reduction of vibration is significant and very similar between the alternatives. Although the GA has not achieved the optimal value, the difference of vibration reduction is only 0.22dB lower.

An important aspect to analyze is the time required to achieve the solution. In order to compare the results with an algorithm which tests all the possibilities, the amount of time needed has been extrapolated from the mean time required to compute an individual in MATLAB with the GA:

$$t_{\text{ind}} = \frac{t_{\text{TOTAL}}}{\texttt{iterations popsize}} = \frac{9413.46\text{s}}{500000\text{it}\ 50\frac{\text{ind}}{\text{it}}} = 3.76\ 10^{-4}\frac{\text{s}}{\text{ind}}. \tag{9.67}$$

The time that it will take to compute all the solutions evaluating the FEM model is also extrapolated. The results are presented in table 9.8.

	GA	MIQP	All combinations	FEM
Combinations tested	$2.5\cdot 10^{6}$	$1.86\cdot 10^{6}$	$2.06\cdot 10^{14}$	$2.06\cdot 10^{14}$
Evaluation time [s]	$3.76\cdot 10^{-4}$	$2.31\cdot 10^{-4}$	$3.76\cdot 10^{-4}$	2.0
Total time [h]	1.19	2.61	$2.15\cdot 10^{7}$	$1.14\cdot 10^{11}$

Table 9.8: Time comparison between optimization options.

These results make it clear that the time required to solve this problem testing all the combinations both with the linear coupling or recomputing the FEM model is totally infeasible while both of the optimization tools proposed in this study achieve optimal (or almost optimal) solutions in a reasonable amount of time. Comparing the times required tot test

all the possibilities with the one required by each one of the alternatives proposed:

$$\frac{t_{\text{TOTAL}_{ap}}}{t_{\text{TOTAL}_{\text{GA}}}} = 9.58\ 10^{10}, \tag{9.68}$$

$$\frac{t_{\text{TOTAL}_{ap}}}{t_{\text{TOTAL}_{\text{MIQP}}}} = 4.37\ 10^{10}, \tag{9.69}$$

these values notice the huge decrease of time in relative terms which can be extended to a different hardware.

10. DVAs in a 3D structure

In this chapter it is discussed the optimization problem introduced in section 6.2 which aim it to find the combination of DVAs and point masses minimize the vibration of some points in a three-dimensional structure.

As explained in section 6.2, this problem is the extension to space of the DVAs in a plate problem discussed previously. Therefore, this chapter will be based in this previous explanation and will permanently cite it in order to avoid repetition and to not extend this text unnecessarily.

In this chapter the problem is formulated in (section 10.1 and then the implementations of the problem with the genetic algorithm and CPLEX are detailed (section 10.2 and section 10.3 respectively). Finally, a case is solved and analyzed as example section 10.4.

10.1 Problem formulation

As in the problem of the DVAs in a plate formulated in section 9.1, there is a structural system, but it is a shell structure or a fuselage in particular.

The same sets of points ($\mathcal{O}$ and $\mathcal{D}$) are defined, as well as the types set ($\mathcal{T}$).

The amount of absorbers placed in the structure (n_d) and the variables defining the type (t_i) and position (p_i) are identical too. The set of points where a device is placed ($\mathcal{J}$) also defined.

However, the **objective** of the problem is extended to take into account the vibration in the three space dimensions,

$$\min\{\sum_{j=1}^{m_o}\sum_{i=1}^{3}(\text{Re}(x_{i_j})^2+\text{Im}(x_{i_j})^2)\}, \tag{10.1}$$

where $x_{1_j}, x_{2_j}, x_{3_j}$ represent the variables x, y, z for a point j, and the $\underline{\mathbf{x}}$ vector is computed solving the there-dimensional linear system from equation 6.11:

$$(\mathbf{I}-\mathbf{H}\ \mathbf{A})\ \underline{\mathbf{X}}=\underline{\mathbf{X}}_0, \tag{10.2}$$

where now $\mathbf{I}$ is the identity matrix of dimension $3(m_o+m_p)$.

$H_{ij} \in \mathbb{C}$ for $i, j = \{1, ..., 3(m_o + m_p)\}$ are the FRFs matrix elements.

$X_0 \in \mathbb{C}$ is a vector which elements represents the vibration of each point $i \in \mathcal{O} \cup \mathcal{D}$ in study in the three space dimension (x, y, z) when there is no device placed in the structure. Its size is $3(m_o + m_p)$.

$X_{\in}\mathbb{C}$ is the unknown vector which elements represents the vibration of each point $i \in \mathcal{O} \cup \mathcal{D}$ in study in the three space dimension (x, y, z) with the new absorber devices configuration. Its size is also $3(m_o + m_p)$.

The matrix $\mathbf{A}$ is built as the ensemble of the rotated equivalent masses belonging to the points where a device is placed (points in $\mathcal{J}$) as defined in equation 6.10.

10.1.1 Genetic algorithm approach

The approach of the problem to the genetic algorithm does not change from the one explained for solving the unidimensional problem in section 9.1.1.

10.1.2 MIQP approach

The problem can also be formulated as a MIQP program so as to be solved with CPLEX Python API. This formulation is very similar to the one presented in section 9.1.2 and only the differences with it are detailed in this section.

Decision variables

The decision variables are exactly the ones defined in section 9.1.2, a total of $(m_o + m_p)$ binary variables for each type of device in $\mathcal{T}$. Besides they are presented in a different way as the same binary variable have to actuate in all x, y, z components for a position, therefore, each set of variables is extended as

$$\underline{\mathbf{Z}}^i = \begin{pmatrix} z_1^i \\ z_1^i \\ z_1^i \\ z_2^i \\ z_2^i \\ z_2^i \\ \vdots \\ z^i_{(m_o+m_p)} \\ z^i_{(m_o+m_p)} \\ z^i_{(m_o+m_p)} \end{pmatrix} \; ; \quad z_j^i \in \{0, 1\}. \tag{10.3}$$

Objective function

The objective function is extended to the three space dimensions with the form presented in equation 10.1,

$$\min\{\sum_{j=1}^{m_o}\sum_{i=1}^{3}(\mathrm{Re}(x_{i_j})^2+\mathrm{Im}(x_{i_j})^2)\}. \tag{10.4}$$

Constraints

The constraints in this problem are also the same presented for the unidimensional problem besides the dynamic relationship (equation 9.15) has to be extended to the space and it becomes equation 10.2.

The steps to follow to transform the dynamic relationship system into a set of linear constraints are the same that in section 9.1.2 but with the extended matrices. Then, it yields

$$\underline{\mathbf{X}}_R-\sum_{i=1}^{t}\left(\left(\mathbf{H_R}\ \mathbf{A_R^i}\ \mathbf{Z^i}\ -\mathbf{H_I}\ \mathbf{A_I^i}\ \mathbf{Z^i}\right)\underline{\mathbf{X}}_R-\left(\mathbf{H_R}\ \mathbf{A_I^i}\ \mathbf{Z^i}+\mathbf{H_I}\ \mathbf{A_R^i}\ \mathbf{Z^i}\right)\underline{\mathbf{X}}_I\right)=\underline{\mathbf{X}}_{0_R} \tag{10.5}$$

for the real part and

$$\underline{\mathbf{X}}_I-\sum_{i=1}^{t}\left(\left(\mathbf{H_R}\ \mathbf{A_I^i}\ \mathbf{Z^i}\ +\mathbf{H_I}\ \mathbf{A_R^i}\ \mathbf{Z^i}\right)\underline{\mathbf{X}}_R+\left(\mathbf{H_R}\ \mathbf{A_R^i}\ \mathbf{z^i}-\mathbf{H_I}\ \mathbf{A_I^i}\ \mathbf{Z^i}\right)\underline{\mathbf{X}}_I\right)=\underline{\mathbf{X}}_{0_I} \tag{10.6}$$

for the imaginary part.

Then, two arrays of variables equivalent to those defined in equation 9.27 and equation 9.28 but extended to three dimensions to be consistent with the dimensions of the problem are defined as

$$\underline{\mathbf{R}}^i=\mathbf{Z}^i\ \underline{\mathbf{X}}_R, \tag{10.7}$$

$$\underline{\mathbf{R}}^i=\mathbf{Z}^i\ \underline{\mathbf{X}}_R. \tag{10.8}$$

The vectors $\underline{\mathbf{R}}^i$ and $\underline{\mathbf{Q}}^i$ contain thee elements for each point in the problem ($j \in \mathcal{O} \cup \mathcal{D}$):

$$\underline{\mathbf{R}}^i = \begin{pmatrix} r^i_{1_1} \\ r^i_{2_1} \\ r^i_{3_1} \\ r^i_{1_2} \\ r^i_{2_2} \\ r^i_{3_2} \\ \vdots \\ r^i_{1_{(m_o+m_p)}} \\ r^i_{2_{(m_o+m_p)}} \\ r^i_{3_{(m_o+m_p)}} \end{pmatrix}, \quad \underline{\mathbf{Q}}^i = \begin{pmatrix} q^i_{1_1} \\ q^i_{2_1} \\ q^i_{3_1} \\ q^i_{1_2} \\ q^i_{2_2} \\ q^i_{3_2} \\ \vdots \\ q^i_{1_{(m_o+m_p)}} \\ q^i_{2_{(m_o+m_p)}} \\ q^i_{3_{(m_o+m_p)}} \end{pmatrix}, \tag{10.9}$$

and introducing them to equation 10.5 and equation 10.6 following linear relationships are obtained:

$$\underline{\mathbf{X}}_R - \sum_{i=1}^{t} \left(\left(\mathbf{H_R}\ \mathbf{A^i_R} - \mathbf{H_I}\ \mathbf{A^i_I} \right) \underline{\mathbf{R}}^i - \left(\mathbf{H_R}\ \mathbf{A^i_I} + \mathbf{H_I}\ \mathbf{A^i_R} \right) \underline{\mathbf{Q}}^i \right) = \underline{\mathbf{X}}_{0_R}, \tag{10.10}$$

$$\underline{\mathbf{X}}_I - \sum_{i=1}^{t} \left(\left(\mathbf{H_R}\ \mathbf{A^i_I} + \mathbf{H_I}\ \mathbf{A^i_R} \right) \underline{\mathbf{R}}^i + \left(\mathbf{H_R}\ \mathbf{A^i_R} - \mathbf{H_I}\ \mathbf{A^i_I} \right) \underline{\mathbf{Q}}^i \right) = \underline{\mathbf{X}}_{0_I}. \tag{10.11}$$

Therefore, as the hypothesis about the variables in these systems presented in section 9.1.2 are still valid, the elements in the vectors $\underline{\mathbf{R}}^i$ and $\underline{\mathbf{Q}}^i$ can be defined with a set of systems of inequations similar to those defined in equations 9.43 to 9.46 for $\underline{\mathbf{r}}^i$ and in equations 9.47 to 9.50 for $\underline{\mathbf{q}}^i$ in the unidimensional problem:

$$r^i_{k_j} \leq B_d\ z^i_{k_j}\ for\ k \in \{1,2,3\},\ j \in \mathcal{O},\ i \in \mathcal{T}, \tag{10.12}$$

$$r^i_{k_j} \geq -B_d\ z^i_{k_j}\ for\ k \in \{1,2,3\},\ j \in \mathcal{O},\ i \in \mathcal{T}, \tag{10.13}$$

$$r^i_{k_j} \leq x_{R_{k_j}} + B_d(1 - z^i_{k_j})\ for\ k \in \{1,2,3\},\ j \in \mathcal{O},\ i \in \mathcal{T}, \tag{10.14}$$

$$r^i_{k_j} \geq x_{R_{k_j}} - B_d(1 - z^i_{k_j})\ for\ k \in \{1,2,3\},\ j \in \mathcal{O},\ i \in \mathcal{T}, \tag{10.15}$$

and

$$q^i_{k_j} \leq B_d \, z^i_{k_j} \; for \; k \in \{1,2,3\}, \; j \in \mathcal{O}, \; i \in \mathcal{T}, \tag{10.16}$$

$$q^i_{k_j} \geq -B_d \, z^i_{k_j} \; for \; k \in \{1,2,3\}, \; j \in \mathcal{O}, \; i \in \mathcal{T}, \tag{10.17}$$

$$q^i_{k_j} \leq x_{I_{k_j}} + B_d(1 - z^i_{k_j}) \; for \; k \in \{1,2,3\}, \; j \in \mathcal{O}, \; i \in \mathcal{T}, \tag{10.18}$$

$$q^i_{k_j} \geq x_{I_{k_j}} - B_d(1 - z^i_{k_j}) \; for \; k \in \{1,2,3\}, \; j \in \mathcal{O}, \; i \in \mathcal{T}. \tag{10.19}$$

10.2 Implementation of the genetic algorithm

The genetic algorithm to solve this problem has been implemented in MATLAB. As explained in section 9.2 this algorithm is made up by generic functions that can be adapted to different problems, therefore most of the functions explained in this previous section are used to solve the three-dimensional problem. In this section the implementation on the genetic algorithm is described but it is focused on the adaptation of the algorithm presented in section 9.2 to the three-dimensional problem.

As the process built to solve the unidimensional problem, it can be divided in two different parts in order to be understood better:

1. Data reading, previous computations and setting up.
2. Genetic algorithm loop.

10.2.1 Data reading, previous computations and setting up

The program starts by defining the constants of the problem and the parameters controlling the algorithm. All of them are the same that has been explained in section 9.2.1.

Regarding the data reading, like in the unidimensional problem; the input force vector, the FRF matrix and the physical properties of the devices must be read. But to solve the 3D problem, the coordinates of the points in $\mathcal{D}$ are required too and they are read from the text file *coordinates.txt*. The format of all these inputs files is detailed in annex C.

Then, the value of the equivalent mass is computed for each device following equation 6.2 and also the rotation angles (θ, ϕ, ψ) are computed. After this, all the functions required by the algorithm are defined with all their constant inputs and, finally, an initial population is generated randomly using the `randomind` function.

10.2.2 Genetic algorithm function

The genetic algorithm used to solve this problem is exactly the same explained in section 9.2.2 since it has been build as a generic function. Moreover, all the strategy functions and the auxiliary functions are also valid since the extension of the problem to the space only changes the fitness function, which is explained in next section.

10.2.3 Fitness function

The fitness function has been developed also as a MATLAB function. Its **inputs** are the same of the unidimensional fitness function (see section 9.2.5) plus

`Th`**:** array containing the rotating angles for all the points in $\mathcal{D}$.

And as an **output** it returns

`fitness`**:** value of the objective function computed by equation 10.1 for the given individual.

The structure of the fitness function for the 3D problem is very similar to the one explained in section 9.2.5 for the unidimensional problem and graphed in figure 9.5.

The first step is to read and decode the given `individual`, while this is done, the equivalent mass matrix is generated in local coordinates following equation 6.7, and the rotating matrix $\mathbf{R}$ (equation 6.9) is computed with the angles from the variable `Th` for that position. Then the equivalent mass matrix is transformed to global coordinates by the rotation defined in equation 6.8, and assembled into matrix $\mathbf{A}$.

Once all the individual is decoded, the right hand side vector $\underline{\mathbf{X}}_0$ is computed and the equations system from equation 10.2 is built. After this, the `cramer` boolean is checked and the system is solved by Cramer's rule or not depending on its value.

Finally, the `fitness` value is computed by equation 10.1 and returned to the genetic algorithm.

10.3 Implementation of the problem in CPLEX

Like the problem regarding DVAs in a plate, this MIQP problem is implemented in the CPLEX Python API. The structure and operation of it is explained in this section.

The application built to solve this problem is very similar to the one built to solve the problem of DVAs in a plate explained in section 9.3. Therefore, it can be divided in the same three parts:

1. Data reading, previous computations and setting up.
2. Definition of the problem.
3. Solution and results processing.

These parts are explained in the next sections focusing in the main differences between them and the ones developed in section 9.3.

10.3.1 Data reading, previous computations and setting up

In this first part of the application, as well as in the one developed in section 9.3.1, all the parameters that define a particular problem are settled, all the information needed is read and all the previous computations are performed.

The parameters defined are exactly the same that the ones for a plane plate in section 9.3.1.

Regarding the information read, like in the plane plate case, the FRF matrices, the device physical properties and the input force vector are required. The reading of this information is formally equal to the one performed in the previous case but it have to be taken into account that the FRF matrices and the input force vector entered to the application must be triple the size of those entered for the unidimensional problem as they must contain the three dimensional coordinates for each one of the points in the problem (points in $\mathcal{O} \cup \mathcal{D}$). Besides, the point coordinates of each node of interest must also be read from the text file *coordinates.txt*. The format of all these files can be found at annex C.

While this data is read, it is computed:

- $\underline{\mathbf{X}}_{0_R}$ and $\underline{\mathbf{X}}_{0_I}$ vectors as indicated by equation 6.5,
- the equivalent masses $m^i_{eq}\ for\ i \in \mathcal{T}$ following equation 6.2,
- the bounds of the variables related to $\underline{\mathbf{x}}$ vector (B and B_d defined in equation 9.31 and equation 9.39),
- the angles $\theta_j, \phi_j \psi_j\ for\ j \in \mathcal{D}$ that constitutes the matrices $\mathbf{R}_j\ for\ j \in \mathcal{D}$ and
- the rotated equivalent masses matrices $\mathbf{M}^i_{eq_j}\ for\ i \in \mathcal{T}\ and\ j \in \mathcal{D}$ following equation 6.8.

10.3.2 Definition of the problem

After reading the required data and performing the previous computations it is defined the MIQP problem in CPLEX.

The first step is to set up the parameters that will be used to solve the problem; it is exactly the same described in section 9.3.2 as it is the same problem but extended to space.

After the setting up, the constraints are populated, and it has been done in three blocks.

The first block works with the real and imaginary parts of the dynamic relationships (equation 10.10 and equation 10.11). First the variables in this equations have to be defined. In this problem there are triple the amount of variables that in the unidimensional one:

- $x_{R_{k_j}}$ is defined for each coordinate ($k = 1, 2, 3$) in each point j in the set $\mathcal{O} \cup \mathcal{D}$.
- $x_{I_{k_j}}$ is defined for each coordinate ($k = 1, 2, 3$) in each point j in the set $\mathcal{O} \cup \mathcal{D}$.
- $r^i_{k_j}$ is defined for each type i in $\mathcal{T}$ and for each coordinate ($k = 1, 2, 3$) in each point j in $\mathcal{D}$. Notice that the variable belonging to the objective points set ($\mathcal{O}$) are not defined as they are always null.
- $q^i_{k_j}$ is defined for each type i in $\mathcal{T}$ and for each coordinate ($k = 1, 2, 3$) in each point j in $\mathcal{D}$. The ones belonging to the set $\mathcal{O}$ are not defined for the same reason explained above with r^i_j.

All of them have their respective bounds, defined by equation 9.31 and equation 9.39.

Then, to build the model in CPLEX, the linear system of equations has to be expanded as explained in section 9.3.2. In order to simplify the indexation of these equations, a new notation is used where vectors $\underline{\mathbf{X}}_R$, $\underline{\mathbf{X}}_I$, $\underline{\mathbf{X}}_{0_R}$ and $\underline{\mathbf{X}}_{O_I}$ components are called $\underline{\mathbf{X}}_{R_j}$, $\underline{\mathbf{X}}_{I_j}$, $\underline{\mathbf{X}}_{0_{R_j}}$ and $\underline{\mathbf{X}}_{O_{I_j}}$ for $i = 1, ..., 3(m_o + m_p)$ respectively. And each one of the sub-matrices making up the overall $\mathbf{A}^i_R$ and $\mathbf{A}^i_I$ matrices is named $\mathbf{A}^{(j)i}_R$ (equation 6.10). $\mathbf{A}^{(j)i}_R$ is a 3x3 matrix containing the components in global coordinates for a device belonging to type i placed in the point j. With this, the linear expressions introduced to CPLEX are:

$$x_{R_j} + \sum_{i=1}^{t} \left[\sum_{k=mo+1}^{mo+mp} \left[\sum_{q=1}^{3} - \left(\sum_{s=1}^{3} \left(H_{R_{j,(3k+s)}} \, A^{(k)i}_{R_{sq}} - H_{I_{j,(3k+s)}} \, A^{(k)i}_{I_{sq}} \right) r^i_{k_j} + \right. \right. \right. \\ \left. \left. \left. + \sum_{s=1}^{3} \left(H_{R_{j,(3k+s)}} \, A^{(k)i}_{I_{sq}} + H_{I_{j,(3k+s)}} \, A^{(k)i}_{R_{sq}} \right) q^i_{k_j} \right) \right] \right] = x_{0_{R_j}} \tag{10.20}$$

for each row j representing the real part and

$$x_{R_j} + \sum_{i=1}^{t} \left[\sum_{k=mo+1}^{mo+mp} \left[\sum_{q=1}^{3} \left(\sum_{s=1}^{3} - \left(H_{R_{j,(3k+s)}} \, A_{I_{sq}}^{(k)i} + H_{I_{j,(3k+s)}} \, A_{R_{sq}}^{(k)i} \right) r_{k_j}^{i} + \right. \right. \right. \\ \left. \left. \left. + \sum_{s=1}^{3} - \left(H_{R_{j,(3k+s)}} \, A_{R_{sq}}^{(k)i} - H_{I_{j,(3k+s)}} \, A_{I_{sq}}^{(k)i} \right) q_{k_j}^{i} \right) \right] \right] = x_{0_{R_j}} \tag{10.21}$$

for each row j representing the imaginary part.

In the second block it is formulated the linear equations systems that defines the relationship between the variables $r_{k_j}^i$ and $q_{k_j}^i$ with the displacement ($x_{R_{k_j}}$ and $x_{R_{k_j}}$ respectively) and the binary variables z_j^i for each space coordinate k of each point j in $\mathcal{D}$ and every type i in $\mathcal{T}$ given by equations 10.12 to 10.19.

In order to implement them in CPLEX, the variables z_j^i must be defined as binary variables and the equations must be properly arranged in the same way that it has been done in section 9.3.2.

In the third block the constraint remaining to define the problem are populated. As they are exactly the same as those in section 9.3.2 (equation 9.16 and equation 9.18) they are not repeated here.

After populating all the constraints, the objective function and its sense are set. The sense of the objective is to minimize and the quadratic objective function is the function presented in equation 10.1, that using the notation defined for this implementation (separeting real and imaginary part of the variables) yields

$$\min\{\sum_{j=1}^{m_o} \sum_{i=1}^{3} (x_{R_{i_j}}^2 + x_{I_{i_j}}^2)\}, \tag{10.22}$$

which, as in section 9.3.2, is equivalent to sum the square value of the variables representing both real and imaginary part of the vibration of every coordinate of the points in $\mathcal{O}$

10.3.3 Solution and results processing

This last part of the application is identical to the one described in section 9.3.3. CPLEX is called to solve the problem and then the solution status, the iterations counter, the objective and the solution reached are printed.

10.4 Case study

This optimization case represents the procedure formulated and explained in this chapter. The geometry for this section has been provided by **SENER Ingeniería y Sistemas** and will be used to develop section 11.3.

10.4.1 Model characterization

The model used for this study is the simplification of a fuselage model with the following components:

- **Skin**: the surface of the fuselage model
- **Frames**: circular reinforcements periodically placed around the longitudinal axis of the fuselage. There is a total number of 5 frames.
- **Stringers**: straight reinforcements periodically placed on the fuselage skin, in the direction of the longitudinal axis. There is a total number of 32 stringers.
- **Floor panel**: plate in which the seats or cargo are placed.
- **Columns**: reinforcement beams joining the floor panel with the fuselage surface. 100mm diameter circular section.

The material that conforms all the components is an aluminum of the properties shown in table 10.1.

$\rho[\frac{kg}{m^3}]$	Poisson	E [GPa]
2796	0.33	73.8

Table 10.1: Aluminum properties for the fuselage model

There are two different boundary conditions, one at the front rim of the fuselage and another one at the rear rim:

- Front: Rigid union of a point mass placed 5 meters in front of the fuselage, in order to simulate in a simple way the effect of the pilot cabin.
- Back: Restriction of all the degrees of freedom of movement and rotation, in order to simulate in a simple way the effect of the rest of the fuselage.

The weight of the passengers is simulated with point masses placed in several points of the floor plate.

The general dimensions of the model are described in table 10.2.

Lenght [m]	Radius [m]	Floor lenght [m]	Floor width [m]	Dist. frames [m]	Thickness [m]
6	1.95	6	3.8	1	0.001

Table 10.2: Fuselage model dimensions in meters

The forces applied to this model simulate the stress transmitted by the engines. They are two antisymmetric forces applied in the outside of the fuselage at the same height of the floor plane and in the middle point of the longitudinal axis. Thet both have a modulus of 1000N.

10.4.2 Nodes of interest

The area that has been selected to be vibration-optimized is the floor panel of the fuselage. Yhe panel in a general overview of the structure model is marked in figure 10.1.

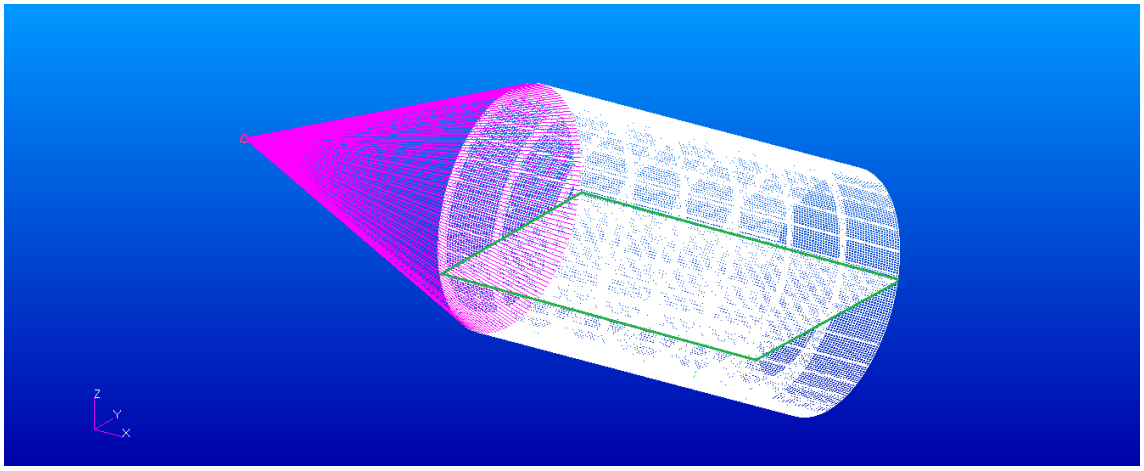

Figure 10.1: Floor section of the fuselage

The points of possible allocation of DVAs or beams have been chosen as the intersection of the horizontal stringers with the circular frames. There is a total of 32 such points in every frame. For reasons of lack of computing power, this study has only been completed considering one of the five frames presented in the model, the central one. The location of the points of possible DVA/beam allocation is displayed in figure 10.2.

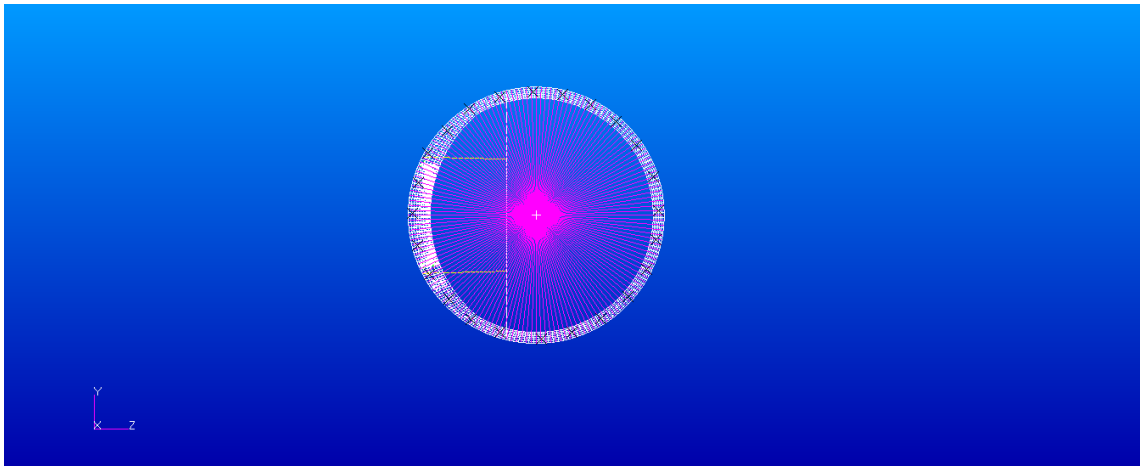

Figure 10.2: Longitudinal view of the structure with possible countermeasure allocation points

10.4.3 Devices characteristics

For this simulation, two kinds of DVA and a point mass have been considered. Table 10.3 represents the physical properties of them.

Type	m [kg]	k [$\frac{\text{N}}{\text{m}}$]	c [$\frac{\text{N s}}{\text{m}}$]	Tuning ω[Hz]
1	0.600	202129.49	9.4876	92.4
2	0.800	202129.49	9.4876	80
3	0.500	∞	0	-

Table 10.3: Properties of the DVAs and the point mass used in the problem.

To represent that the third types is a point mass a rigidity value with the code 999999999 is given to it (see annex C).

10.4.4 Preprocessor

The preprocessor consists in computing the FRF matrix for the interest nodes by means of NASTRAN. The methodology followed in this case is the extension of the one explained in section 9.4. For a detailed explanation see [2].

10.4.5 Optimization by the genetic algorithm

In order to solve this problem with the implemented GA, it is need to create the inputs files containing:

- the real part of the FRF matrix,
- the imaginary part of the FRF matrix,
- the coordinates of the points of interest,
- The input force vector.
- the physical properties of the available DVAs,

and these files must follow the format defined in annex C.
Once the files are ready, the constants defining the problem have to be given to the algorithm. It is done assigning the values presenter in table 10.4 to the variables of the program.

types	3
ndmax	20
positions	32
op	20
w	80

Table 10.4: GA constants of the problem.

Then, the parameters of the algorithm have to be set up. As the aim of this case is to illustrate the proper operation of the algorithm but not to optimize its performance their discussion is out of the scope of this study. They will be the same used in section 9.4 and presented in table 9.4.

After performing the 500000 iterations in 70166 s, the best individual achieved is presented in table 10.5. It has a fitness value of $1.44189 \cdot 10^{-5}$ m^2.

This solution is equivalent to an insertion loss of

$$IL_{GA} = 3.36 dB,$$

and it supposes an increase of 13.6kg.

10.4.6 Optimization by MIQP

The optimization tool developed to call the CPLEX solver needs as input the same files that the GA (FRF matrix, input force, DVA properties and point coordinates).

Then, the main variables of the project have to be given to the application (see section 9.3). They are presented in table 10.6.

Type	Node ID
2	1
1	5
1	6
1	7
1	8
1	11
1	13
1	15
1	16
2	17
1	18
1	19
2	20
1	21
2	22
1	23
2	26
2	28
2	29
2	30

Table 10.5: Best individual achieved with the GA.

The solution achieved is the optimal solution and is presented in table 10.7. It has been codified as a GA individual in order to compare both solutions.

The objective value reached is $1.48522 \cdot 10^{-5}$ m^2, which represents an insertion loss of

$$IL_{MIQP} = 3.23 \text{dB}.$$

This solution has been achieved after $9.260 \cdot 10^7$ iterations in a time of 79419s and it supposes an increment of 13.4kg. However, with this time and this number of iterations CPLEX doesn't ensure to find the optimal solution; it is clear when compared with the one achieved with GA, which has a lower objective value.

The time required to ensure that the solution in an optimum for this problem is too big for the available computational resources, then, in order to help the solver it is introduced the solution found by the GA (table 10.5) as a presolution to see if CPLEX is able to improve it and find a better one. Nevertheless, after $1.90 \cdot 10^8$ iterations performed in 169620s it was not.

mo	20
mp	32
t	3
nd_max	20
w	80

Table 10.6: Variables of problem introduced to MIQP application.

10.4.7 Analysis of results

The fist to notice is that the insertion loss achieved for both of the optimization tools is very similar and that CPLEX is not able to reduce the solution achieved with GA. This indicates that both alternatives achieve a close-to-optimal solution without many troubles and also that the MIQP program has a real problem of scalability and when the problem grows, like in this example with 96 binary variables and 216 real variables, making it infeasible to ensure that the solution is a the optimum one at least with the time and resources available.

Something that might call the attention of the reader is that the best solution found is a mix of devices and not all of them are tuned. But if it is tested a solution with the devices placed by the best solution found (table 10.5) but with all the DVAs tuned (type 2) the reduction achieved is only 1.70dB. This shows that in complex structures like this the best solution may not be intuitive at all.

The increase of weight that these solutions imply can be computed in terms relative to the total weight of the whole structure:

$$\Delta w_{GA} = \frac{13.6\text{kg}}{1756\text{kg}} = 0.774\% \tag{10.23}$$

$$\Delta w_{\text{MIQP}} = \frac{13.4\text{kg}}{1756\text{kg}} = 0.763\% \tag{10.24}$$

These values show that the increase of weight is negligible compared to the weight of the fuselage for the reduction achieved. When the reduction achieved (3.356dB) is compared with this increase of weight (0.774%), it can be concluded that the reduction has a significant value.

In this case, the GA has achieved a higher vibration reduction in a similar amount of time but also with a higher increase of mass. This highlights the fact that the algorithms look for the minimum level of vibration without taking into account any weight parameter. The task of the engineer in here is to choose between these solutions (and maybe also some others close-to-optimial solutions that the algorithms have test before finishing) consider-

Type	Node ID
2	1
2	4
1	7
3	8
1	10
1	12
2	13
1	14
1	16
2	17
3	18
2	20
2	22
3	23
1	24
1	25
2	26
2	28
2	30
3	31

Table 10.7: Solution achieved by MIQP.

ing more parameters than the vibration reduction as can be the increase of weight or the cost of implementing the solution.

Another important aspect to analyze is the time required to achieve the solution. In order to compare the results with an algorithm which tests all the possibilities, the amount of time stipulated has been extrapolated like in section 9.4:

$$t_{\text{ind}} = \frac{t_{\text{TOTAL}}}{\text{iterations popsize}} = \frac{70166\text{s}}{500000\text{it } 50\frac{\text{ind}}{\text{it}}} = 2.81 \cdot 10^{-3}\frac{\text{s}}{\text{ind}}. \tag{10.25}$$

The time that it will take to compute all the possibilities is not contemplate as it can easily seen that it is absolutely infeasible.

The results are presented in table 10.8.

These results make it clear that the time required to solve this problem testing all the combinations is totally infeasible while both of the optimization tools proposed in this study achieve close-to-optimal solutions in a reasonable amount of time.

Comparing the time required to test all the possibilities with the one required by each one

	GA	MIQP	All combinations
Combinations tested	$2.5{\cdot}10^{7}$	$9.26{\cdot}10^{7}$	4 $1.15{\cdot}10^{18}$
Evaluation time [s]	$2.81{\cdot}\ 10^{-3}$	$8.577{\cdot}10^{-4}$	$2.81{\cdot}10^{-3}$
Total time [h]	22.06	16.23	$9.0{\cdot}10^{11}$

Table 10.8: Time comparison between optimization options.

of the alternatives proposed:

$$\frac{t_{\text{TOTAL}_{\text{ap}}}}{t_{\text{TOTAL}_{\text{GA}}}} = 4.04 \cdot 10^{10}, \tag{10.26}$$

$$\frac{t_{\text{TOTAL}_{ap}}}{t_{\text{TOTAL}_{\text{MIQP}}}} = 5.54 \cdot 10^{10}, \tag{10.27}$$

these values notice that the time of the solution hugely reduce the computation time, being about ten orders of magnitude faster.

11. Stiffeners and DVAs in a 3D structure

The last extension of the problem is presented in this chapter. It consist in adding stiffeners to the structure too, instead of just DVAs.

In the next sections the problem is formulated, in section 11.1, then the implementation of it with the genetic algorithm is detailed in section 11.2 and finally a case is solved and analyzed in section 11.3.

In order to implement an optimization process to solve this problem some hypothesis has to be taken:

- Stiffeners will be modeled by beams connecting two consecutive points.
- The beams are formulated as Euler's beams.
- The beams only can be placed in a plane perpendicular to the longitudinal axis.
- The points where the devices can be placed must be chosen counterclockwise and by planes perpendicular to the longitudinal axis.
- There must be the same amount of points for each of the planes where a device can be placed.

11.1 Problem formulation

As in the previous problems, there is a structural system, a shell structure or an airplane fuselage in particular. In this structure a set of objective points ($\mathcal{O}$) and a set of possible points where the different devices can be placed ($\mathcal{D}$) are defined as in section 9.1.

The set of types $\mathcal{T}$ is also defined, but in this case it is formed by the union of two sets:

- Set of DVA types: set of natural numbers coding the different types of DVA that can be placed in the structure,

$$\{DVA\ Types\} = \mathcal{T}_{\mathcal{D}} \in \mathbb{N}. \tag{11.1}$$

- Set of stiffeners types: set of natural numbers coding the different types of DVA.

The lower element in this set is the next natural after the higher element in $\mathcal{T}_\mathcal{D}$,

$$\{Stiffness\ Types\} = \mathcal{T}_\mathcal{S} \in \mathbb{N}. \tag{11.2}$$

Hence, the set of types is defined as

$$\mathcal{T} = \mathcal{T}_\mathcal{D} \cup \mathcal{T}_\mathcal{S}. \tag{11.3}$$

The amount of devices placed in the structure is still n_d; it now contains both stiffeners and DVAs. Each device placed is defined by its types (t_i) and its position (p_i) as defined in section 9.1.

The set of points where a device is placed is still defined as $\mathcal{J}$ but like $\mathcal{T}$, it is now formed by the union of two sets:

- Set of points where a DVA is placed:

$$\mathcal{J}_\mathcal{D} = \{p_i \mid t_i \in \mathcal{T}_\mathcal{D}\ and\ i = 1, ..., n_d\}\ \ \mathcal{J}_\mathcal{D} \in \mathcal{D}. \tag{11.4}$$

- Set of points where a DVA is placed:

$$\mathcal{J}_\mathcal{S} = \{p_i \mid t_i \in \mathcal{T}_\mathcal{S}\ and\ i = 1, ..., n_d\}\ \ \mathcal{J}_\mathcal{S} \in \mathcal{D}. \tag{11.5}$$

Hence, the set of all the points containing a device is

$$\mathcal{J} = \mathcal{J}_\mathcal{D} \cup \mathcal{J}_\mathcal{S}. \tag{11.6}$$

Notice that it is considered that there cannot be more than one device in a single point, not even if they are from a different nature, hence

$$\mathcal{J}_\mathcal{D} \cap \mathcal{J}_\mathcal{S} = \emptyset. \tag{11.7}$$

The **objective** of the problem is exactly the same define for the problem of DVAs in a 3D structure in equation 10.1 in section 10.1,

$$\min\{\sum_{j=1}^{m_o}\sum_{i=1}^{3}(\mathrm{Re}(x_{i_j})^2 + \mathrm{Im}(x_{i_j})^2)\},$$

where $x_{1_j}, x_{2_j}, x_{3_j}$ represent the variables x, y, z for a point j, and the $\underline{\mathbf{x}}$ vector is the computed solving the there-dimensional linear system from equation 6.17 in section 6.3:

$$(\mathbf{I} + \mathbf{H}\,(\mathbf{H}_B - \mathbf{A}))\,\underline{\mathbf{X}} = \underline{\mathbf{X}}_0, \tag{11.8}$$

where now $\mathbf{I}$ is an identity matrix of dimension $6(m_o + m_p)$,

$H_{ij} \in \mathbb{C}$ for $i, j = \{1, ..., 6(m_o + m_p)\}$ is the FRFs matrix,

$X_0 \in \mathbb{C}$ is a vector which elements represents the vibration of each point i in study in the three space dimension (x, y, z) and the three respective twists with $i \in \mathcal{O} \cup \mathcal{D}$ when there is no device placed in the structure. Its size is $6(m_o + m_p)$.

$X_{\in}\mathbb{C}$ for is the unknown vector which elements represents the vibration of each point i in study in the three space dimension (x, y, z) and the three respective twists with $i \in \mathcal{O} \cup \mathcal{D}$ with the new absorber devices configuration. Its size is also $6(m_o + m_p)$.

The matrix $\mathbf{A}$ is build as the ensemble of the rotated equivalent masses belonging to the points where a device is placed ($\mathcal{J}$) as defined in equation 6.10 extended to six dimensions as it has been explained in section 6.3, and the matrix $\mathbf{H}_B$ is build with both the assembled mass and stiffness matrices of the points where a stiffener is placed ($\mathcal{J}_\mathcal{S}$) as defined in equation 6.14.

11.1.1 Genetic algorithm approach

The approach of the problem to the genetic algorithm does not change from the one explained for solving the problems where only DVAs are taken into account (section 9.1.1 and section 10.1.1) .

The discernment between stiffeners and bars is captured in each type variable and it will be treated inside the `fitness function` (section 11.2.4).

11.2 Implementation of the genetic algorithm

The problem regarding stiffeners and DVAs in a three-dimensional structure has been solved by the genetic algorithm used in the previous problems.

As much functions as possible are taken from the previous cases, in particular from the 3D problem. Therefore, in order to not unnecessarily lengthen this text, in this section it will be detailed just the main difference with the genetic algorithm implemented in section 10.2.

In order to understand better how the algorithm works, it can be divided in two different parts as it has been done in section 9.2 and section 10.2:

1. Data reading, previous computations and setting up.

2. Genetic algorithm loop.

11.2.1 Data reading, previous computations and setting up

The program starts defining the constants of the problem and the parameters controlling the genetic algorithm. In this part two changes are introduced. The first one is to define the two variable that will allow the program to discern between DVAs and stiffeners when dealing with an individual:

`types_DVA`**:** number of different types of DVA that will be in the problem (including point masses). It corresponds with the size of the set $\mathcal{T}_{\mathcal{D}}$.

`types_BEAMS`**:** number of different types of beams that will be in the problem. It corresponds with the size of the set $\mathcal{T}_{\mathcal{S}}$.

Therefore, the variable `types`, which corresponds with the size of $\mathcal{T}$, is now the total amount of different devices that can be placed in the structure and it is expressed as the sum of `types_DVA` and `types_BEAMS`.

The last change is to define another new variable:

`nodes_plane`**:** amount of nodes belonging to $\mathcal{D}$ for plane in study (see hypothesis of the problem).

Regarding the reading of data, like in the problem with just DVAs, the input force, the FRF matrix, the physical properties of the devices and the coordinates of the points in $\mathcal{D}$ must be read. In order to deal with stiffeners, the properties of the beams are also required to compute their mass and stiffness matrices (see section 6.3). This data is read from the file *beam_properties.txt* with the format presented in annex C.

After reading all the information, the value of the equivalent mass is computed for each device following equation 6.2 and also the rotation angles (θ, ϕ, ψ) are computed as in section 10.2. Moreover, in order to properly treat the stiffeners, the length of each possible beam and its rotation angles in the plane (θ_B) must be computed too.

11.2.2 Genetic algorithm function

The genetic algorithm used to solve this problem is exactly the same explained in section 9.2.2 since it has been build as a generic function. Moreover, all the strategy functions and the auxiliary functions are also valid since this extension of the problem just changes the fitness function and adds an auxiliary function to compute the mass and stiffness matrices of the bars in an individual which is only called by the fitness function itself. Both of them are explained in the next sections.

11.2.3 Auxiliary functions

11.2.3.1 Mass and stiffness matrices function

This is a MATLAB function which computes both mass and stiffness elemental matrices of each beam that is placed in a given configuration.

The **inputs** required are:

`Length`**:** array containing the length of all the possible beams in the structure.

`Properties`**:** matrix containing the properties of all the possible beams in the structure.

`ThB`**:** array of beam rotation angles (θ_B).

`T`**:** current individual.

`mp`**:** amount of different positions in which a device can be placed.

`typesDVA`**:** number of different types of DVA that will be in the problem (including point masses).

And it returns as **outputs**:

`K`**:** array of matrices containing all the elemental stiffness matrices for the individual given.

`M`**:** array of matrices containing all the elemental mass matrices for the individual given.

This function reads the individual and computes the stiffness and mass matrices for each device which type code indicates that is a stiffener. The matrices are computed in local coordinates following the formulation presented in annex A, then they are transformed to global coordinates by means of a rotation and returned as `M` and `K`.

11.2.4 Fitness function

The fitness function for this problem has been build as a function in MATLAB. This one is a complex function which has to discern between DVAs and stiffeners and to treat them accordingly.

In order to simplify this function it has been built in such a way that an auxiliary function which computes the $\mathbf{K}$ and $\mathbf{M}$ matrices has to be called before calling the fitness function itself. This auxiliary function is described in section 11.2.3.1.

The **inputs** required are the same described in section 9.2.5 plus

`Th`**:** array containing the rotating angles of the DVAs for all the point in $\mathcal{D}$.

`types_DVA`**:** number of different types of DVA that will be in the problem (including point masses).

`K`**:** array of matrices containing all the elemental stiffness matrices for the individual given.

`M`**:** array of matrices containing all the elemental mass matrices for the individual given.

`nodes_plane`**:** amount of nodes belonging to $\mathcal{D}$ for plane in study.

The **output** returned is

`fitness`**:** value of the objective function computed by equation 10.1 for the given individual.

The basic structure of this function is represented in figure 11.1. The first step is to decode the individual. For each row (i) of the individual it is checked if the variable indicating the type of the device to place (t_i) is lower or equal than `types_DVA`. If it is, it means that the device is a DVA, if not that it is a stiffener.

If the device is a DVA it is treated as explained in section 10.2.3 and its equivalent mass matrix is assembled to the global $\mathbf{A}$ matrix.

Otherwise, if it is a stiffener, the first step is to compute the second node of the beam. This node will be the next node on the codification except if it is the last node in a plane, then it will be the first node in that plane (see hypothesis). Then the elemental mass and stiffness matrices connecting these nodes are taken from the variables `M` and `K` and are assembled into the respective global matrices ($\mathbf{M}_G$ and $\mathbf{K}_G$).

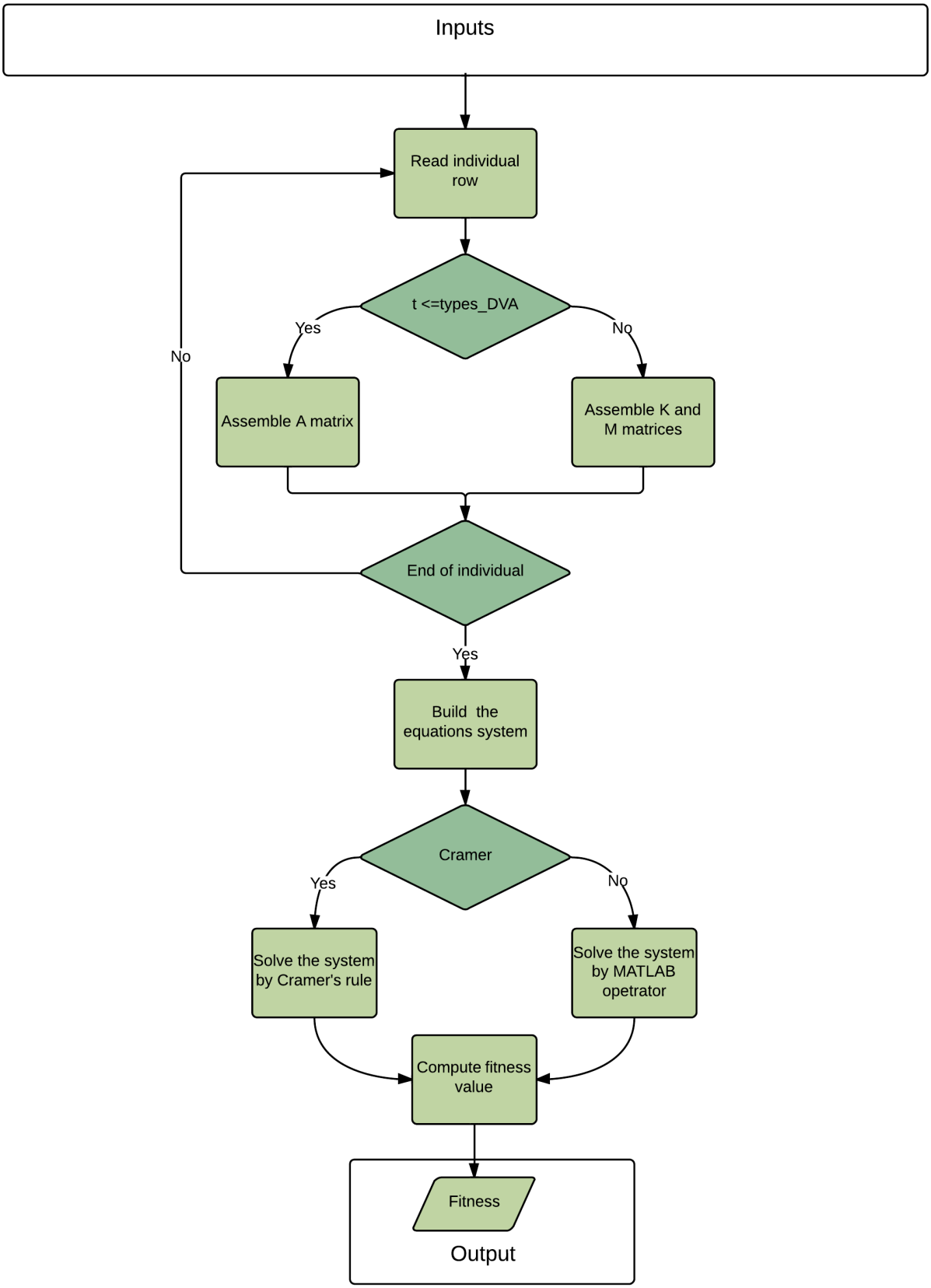

Figure 11.1: Fitness function diagram

O nce

all the individual rows are read and treated accordingly, the right hand side vector ($\underline{\mathbf{X}}_0$) is computed and the system of equations presented in equation 6.15 is built.

The next steps are similar to those explained previously for those functions regarding only DVAs. The `cramer` boolean is checked and the system is solved accordingly.

Finally the `fitness` value is computed by equation 10.1 and returned to the main loop of the algorithm.

11.3 Case study

To the study of this case it is used the structure from section 10.4. The points of interest and the devices are also the same explained in that section, the only difference is that in this case it is possible to place also two different kinds of beams.

11.3.1 Beam characteristics

In the Beam+DVA study two kinds of beam have been considered. Beam kind 1 is made of iron and beam kind 2 is an aluminum alloy. The physical properties and dimensions of the two kinds of beams are presented in table 11.1. The parameters marked with a dash

Types	E [Pa]	G [Pa]	I_y[m^4]	I_z[m^4]	A [m^2]	L [m]	J [m^4]	ρ [$\frac{kg}{m^3}$]
4	$211{\cdot}10^9$	$82{\cdot}10^9$	$2.13{\cdot}10^{-7}$	$2.13{\cdot}10^{-7}$	0.0008	-	$3.60{\cdot}10^{-7}$	7874
5	$70{\cdot}10^9$	$82{\cdot}10^9$	$3.41{\cdot}10^{-6}$	$3.41{\cdot}10^{-6}$	0.0008	-	$3.60{\cdot}10^{-7}$	2700

Table 11.1: Physical properties of the types of beams.

are irrelevant for this set-up.

11.3.2 Preprocessor

As it is the same structure that in section 10.4 the preprocessor performed there is totally valid adding the twist components to the FRF matrix.

11.3.3 Optimization by the genetic algorithm

In order to solve this problem with the implemented GA several input files have to be generated:

- the real part of the FRF matrix,

- the imaginary part of the FRF matrix,
- the coordinates of the points of interest,
- the input force vector,
- the physical properties of the available DVAs and
- the physical properties of the available beams.

These files must follow the format defined in annex C.

Once the problem is defined, the constants defining it have to be given to the algorithm. They are presented in table 11.2.

types_DVA	3
types_BEAMS	2
ndmax	10
positions	32
op	20
nodes_planes	32
w	80

Table 11.2: GA constants of the problem.

Then, the parameters of the algorithm have to be set up. As in the previous cases, their discussion is out of the scope of this study and they will be the same used in the one dimensional case (see section 9.4) but the population size which is reduced to 30 individuals for computing time reasons.

After performing 500000 iterations in 166166s, the best individual achieved is presented in table 11.3, which has a fitness value of $3.1588 \cdot 10^{-6}$ m^2.

This solution represents an insertion loss of

$$\text{IL} = 9.95\text{dB}$$

with an increase of weight of 16.0kg.

11.3.4 Analysis of results

The insertion loss achieved for this case is almost a reduction of one order of magnitude in the vibration level. A physical analysis of the solution shows that the structure analyzed has a lack of stiffness as all the devices placed are beams. It also can be noticed that the solution places the beams in the first half of the points. This makes sense because this

Type	Node ID
5	1
5	2
5	3
4	6
5	7
4	8
4	9
4	12
5	14
5	16

Table 11.3: Best individual achieved with the GA.

positions are the ones under the floor, where the floor itself and the columns that support it are connected with the frame, therefore stiffening the structure in that zone will clearly reduce the vibration in the floor.

The increase of weight that these solutions imply can be computed in terms relative to the total weight of the plate:

$$\Delta w_{GA} = \frac{16.0\text{kg}}{1756\text{kg}} = 0.91\%. \tag{11.9}$$

This increase is not significant taking into account the hight reduction that has been achieved which gives even more value to the solution found.

Finally, the time required to achieve the solution is analyzed. In order to compare the results with an algorithm which tests all the possibilities, the amount of time stipulated has been extrapolated like in section 10.4:

$$t_{\text{ind}} = \frac{t_{\text{TOTAL}}}{\text{iterations popsize}} = \frac{166166\text{s}}{500000\text{it } 30\frac{\text{ind}}{\text{it}}} = 1.11 \cdot 10^{-2}\frac{\text{s}}{\text{ind}}. \tag{11.10}$$

The time that it will take to compute all the possibilities is not contemplate as it can easily seen that it is absolutely infeasible.

The results are presented in table 11.4.

These results make it clear that the time required to solve this problem testing all the combinations is totally infeasible while the optimization tool proposed in this study achieve a close-to-optimal solution in a reasonable amount of time.

	GA	All combinations
Combinations tested	$1.5{\cdot}10^{7}$	$7.96{\cdot}10^{24}$
Evaluation time [s]	$1.11{\cdot}\ 10^{-2}$	$1.11{\cdot}10^{-2}$
Total time [h]	46.16	$6.82{\cdot}10^{17}$

Table 11.4: Time comparison between optimization options.

Comparing the time required to test all the possibilities with the one required by the GA:

$$\frac{t_{\text{TOTAL}_{\text{ap}}}}{t_{\text{TOTAL}_{\text{GA}}}} = 1.48 \cdot 10^{16}, \tag{11.11}$$

this value notice that the time has been hugely reduced.

12. Safety

The world of airspace engineering has a big challenge in assuring the safety of the users of its technology, since failures can greatly affect the success of the industry in the eyes of public opinion.

The fatigue created by high levels of vibration negatively affects the integrity of structural materials, favoring the appearance of cracks or even their sudden collapse. Airplanes can experience the formation of cracks in specifically challenged areas of their structure, such as the engine support or the wing fuselage union, such as the recent issues with the wings of the Boeing 787 [20].

The tool developed in this study has the potential of preventing high levels of vibration, thus reducing the risk of formation of cracks in critical areas of an airplane fuselage and so guaranteeing the integrity of the vehicle.

13. Environmental implication

Modern engineering is not only required to accomplish objectives of functionality and economical aspects, but also of environmental impact. This is the case especially in those industries that develop and manufacture products which emit substances during their normal working cycle that can potentially harm the environment.

In recent years, the aerospace industry has invested a high amount of resources in the development of fuel efficient and eco-friendly technologies. One such effort has been the development of alternative fuels for NASA rockets, thus discarding the extremely hazardous solid rocket engines.

Another challenge that the aerospace companies have to overcome is the increasing acoustic pollution generated by the ever-growing fleets of commercial airplanes.

In a smaller impact scale, the simulations used for aerospace applications can reach prohibitive levels of electricity consumption by the required hardware. When these values are pushed to the limit of feasibility, companies and research centers end up consuming very high amounts of electricity to perform their simulations. Physical and mathematical models that can help reduce the amount of calculations have a direct effect on power consumption.

13.1 Fuel emissions reduction

There is a great dependency between total weight of an aircraft and fuel consumption. Any technology that makes part of the mass of an airplane expendable directly reduces the fuel consumption of the vehicle, which leads to a reduction of emissions.

An aircraft that includes countermeasures for reduction of vibration levels can undergo the procedure of optimization, achieving the same result as the original configuration with a significant weight reduction.

13.2 Acoustic pollution reduction

A reduction of certain noise levels can be achieved with the correct placement of vibration dissipating elements. If the optimization cycle is tuned to eliminate certain frequencies of vibration, the outcome can greatly reduce the noise generated by the vibration of the fuselage components in that frequency.

Acoustic countermeasures can be greatly effective if tuned against vibration frequencies generated in take-off and landing operations, greatly improving the comfort of nearby residential zones.

The reduction of cruise flight acoustic pollution would allow access to flight paths over protected natural or residential zones, allowing airplanes to cross these zones without significantly perturbing the levels of noise.

13.3 Reduction of simulation energetic cost

The optimization cycle proposed in this study acts on simulation efficiency in two main aspects:

- **Linearized problem:** A linearization of the problem allows the method to avoid the re-calculation of the whole structure through a finite element simulation, with very low error, which would lead to unnecessary computation time and energy consumption
- **Optimization techniques:** The addition of optimization techniques greatly reduces the amount of possible configurations that have to be analyzed in order to obtain the optimal solution. The calculation of far-to-optimal solutions is irrelevant for the purpose of the method so avoiding it reduces energy consumption without any effect on the final result.

14. Limitations and future lines of investigation

The algorithms presented in this study are definitively a good alternative to optimizing the absorber devices in an airplane structure, nevertheless, they still have some limitations which can be developed in future studies.

- The algorithms have been tested and they hugely improve the efficiency of the optimization.This make feasible to analyze a great amount of combinations in a relatively little period of time obtaining very good results or even optimal results. However, in this study the influence of the different parameters controlling the performance of the algorithms has not been analyzed. It will be a good point to make this study in order to improve the efficiency of the optimization processes for a particular case or even for a group of similar cases such as airplane fuselages.
- The algorithms have been tested in some illustrative cases in order to prove that they are very useful and powerful tools. It will be a good experience to go further and test the behavior of the algorithms in bigger problems with better computational resources in order to study their scalability, their real power and their limitations. For these cases it may be interesting to test the combination of both algorithms, trying to improve a result from the genetic algorithm with the MIPQ application, for example.
- After testing the limitations of the implemented algorithm, an interesting study will be to use different algorithms that apparently are not the best choice, for example nonlinear integer programming, and analyze if they achieve a better behavior in problems with a bigger amount of data to process.
- To improve the efficiency of the algorithms, different methods to take advantage of symmetries or the repeatability of some structural elements as formers may be studied and implemented.
- The optimization presented in this study is focused only in a single working frequency. A good improvement may be to perform the optimization not only on a single frequency but on a frequency window around the one of interest or even on

multiple windows around this frequency and its harmonics. The implementation of this new feature is almost direct if the range of frequency is discretized.

- The analysis and optimization developed in this study stays in the field of the vibration reduction. The next natural step would be to extend it in terms of acoustic radiation.

- The objective of the optimization performed in this study is a pure minimization of the vibration. However, it may be interesting for some practical cases, as can be an aircraft fuselage, to introduce in the objective function some terms related with the increase of weight or even with economic aspects. This would make the problem not look for a pure minimum in vibration but achieve an agreement between the vibration reduction and these new variables.

- The genetic algorithm has been implemented in MATLAB. With this language it is usually faster to develop and perform programs or algorithms than with traditional programming such as C or C++. Nevertheless, it is definitively not more efficient in pure terms of run time. It may be a good improvement to convert the code to a lower-level programming language such as C++. Regarding the CPLEX Python API used to solve the MIQP, the same reasoning can be applied. Python is a very comfortable language and it allows an easier and faster approach to the CPLEX tool but the run time could be reduced converting the application to a lower-level programming language.

- The formulation of the DVAs in a plate has been already experimentally validated in [1] but the stiffeners model has not. A future study should test it experimentally implementing the solution given by the algorithms and validating the results obtained.

- The devices treated in this study are traditional models of DVA, with parallel spring and damping, and Euler's beams for stiffeners. A possible future study may be to test new models formulated for example with Timoshenko beams theory or with alternative models of DVA such as series spring and damping models and test them experimentally.

15. Planning and scheduling

In this section it is described the planning and scheduling followed to develop the project (section 15.1) and a possible future planning to continue with the study following the future lines of investigation (see chapter 14) is presented (section 15.2).

15.1 Planning and scheduling followed

The planning and scheduling followed to develop a project is detailed in this section. No significant scheduling problems have been presented during the study and only normal rescheduling tasks have been done during this time in order to introduce new tasks and to solve some unexpected events.

The description of the tasks followed is presented above and the Gantt chart is in figure 15.1.

1.Information review and research: review and research of information related with the project.

- **1.1.Review of vibro-acoustic theory:** review of the basic vibro-acoutic theory to be applied in the project.
- **1.2.Review of existing methodology:** review of the computationally efficient formulation that will be used to solve the problem. Information provided by SENER.
- **1.3.Information research:** research for other methodologies or similar projects.

2.Identification of the problem: identification of the mathematical problem and the functions to be optimized.

3.Evaluation of the best optimization algorithms: evaluation of the best possible alternatives to solve the problem.

4.DVAs in a rectangular plane plate: development of the process applied to a rectangular plate concerning just DVAs and point masses.

- **4.1.Formulation of the unidimensional problem:** formulation of the optimization problem for the unidimensional case.

4.2.Set up and FRF computation: development of the Nastran input file and the script that allows the user to obtain the FRF matrix.

4.3.Implementation of the optimization tools (1D): implementation of the proper tools to optimize the problem with the algorithms chosen.

4.4.Adaptation of the algorithm to the vibro-acoustic design cycle: coupling of the algorithms with the vibro-acoustic solving software (linear formulation).

4.5.Case study: Optimization of an illustrative case and validation or discarding of the results obtained following physical criteria .

5.DVAs in a 3D structure: development of the process applied to a semimonocoque structure concerning just DVAs and point masses for a plane plate.

5.1.Formulation of the 3D problem: formulation of the optimization problem for the three-dimensional case concerning just DVAs and point masses.

5.2.Set up and FRF computation: Development of the Nastran input file and the script that allows the user to obtain the FRF matrix of the 3D structure.

5.3.Implementation of the optimization tools (3D): implementation of the proper tools to optimize the problem with the algorithms chosen.

5.4.Case study: Ootimization of an illustrative case and validation or discarding of the results obtained following physical criteria for an airplane structure where DVAs and point masses can be used as countermeasures.

6. Beams DVAs in a 3D structure: development of the process applied to a semimonocoque structure adding beams as a possible vibration countermeasure.

6.1.Formulation of the problem: formulation of the optimization problem for the three-dimensional case concerning beams, DVAs and point masses.

6.2.Refinement of the GA: refinement of the genetic algorithm in order to allow the treatment of beams.

6.3.Case study: optimization of an illustrative case and validation or discarding of the results obtained following physical criteria for an an airplane structure where beams, DVAs and point masses can be used as countermeasures.

7.Document drafting: creation of the report and annexes that describe the development of the project as well as the final results.

15.1.1 Schedule

Name	Work
Information review and research	**11d**
Review of vibro-acoustic theory	3d
Review of existing methodology	3d
Information research	5d
Identification of the problem	2d
Evaluation of the best optimization algorithms	2d
DVAs in a rectangular plane plate	**32d**
Formulation of the unidimensional problem	2d
Set up and FRF computation	4d
Implementation of the optimization tools (1D)	15d
Adaptation of the algorithm to the vibro-acoustic design cycle	7d
Case study and analysis of results	4d
DVAs in a 3D structure	**21d**
Formulation of the 3D problem	2d
Set up and FRF computation	4d
Implementation of the optimization tools (3D)	10d
Case study and analysis of results	5d
Beams and DVAs in a 3D structure	**11d**
Formulation of the problem with beams	2d
Refinement of the GA	4d
Case study and analysis of results	5d
Documents drafting	20d

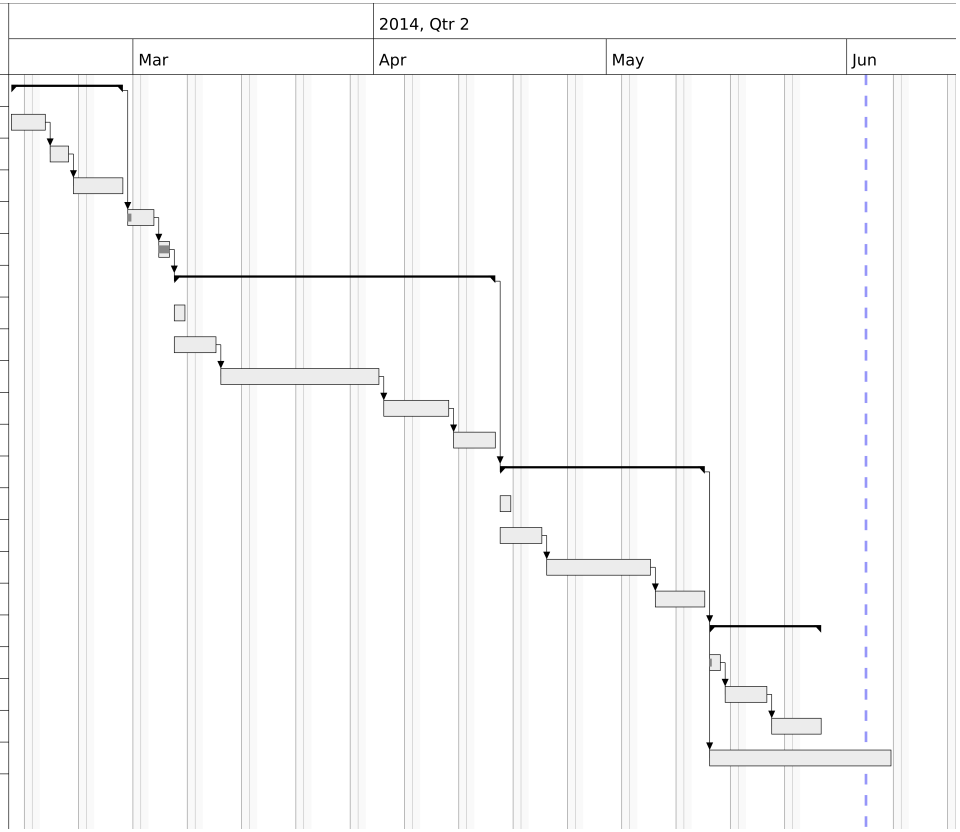

Figure 15.1: Gantt chart followed to develop the study.

15.2 Future planning and scheduling

In this section it is proposed a planning and scheduling to continue with the development of the study in order to fulfill some of the aspects presented in section 14. Like this study, the future steps are thought to be developed in parallel with an assumed continuation of the study developed by D.Sellés in [2].

The description of the proposed tasks is presented below and a possible schedule starting on 1st September 2014 is graphed as a Gantt chart in figure 15.2.

1. **Adaptation of the optimization tools to frequency windows:** make the proper changes in the already developed optimization tools in order to allow an optimization in various frequencies.

2. **Study of the effects of GA parameters in efficiency:** study how the parameters that controls the performance of the GA affect to the efficiency of the method and its convergence.

   2.1. **Study of mutation ratio effect:** study of the effect of the mutation ratio parameter in the efficiency of the GA.

   2.2. **Study of reproduction ratio effect:** study of the effect of the reproduction ratio parameter in the efficiency of the GA.

   2.3. **Study of elitism effect:** study of the effect of the size of elite population in the efficiency of the GA.

3. **Optimization of real-size problems:** preparation and solution of a problem of real size in order to determine the scalability of the optimization tools developed in this study.

   3.1. **Preprocessor:** preparation of the case, choice of interesting points and computation of FRF matrix.

   3.2. **Solution and analysis of results:** solution of this real-size problem and analysis of results in order to know the scalability of the algorithms.

4. **Test new optimization algorithms:** test of new alternative algorithms that could give good results in huge problems despite not being the most suitable solution.

   4.1. **Evaluation of the possible alternatives:** research about possible algorithms that could have a good behaviour with huge problems despite not being the best solution a priori.

4.2.Formulation of the new models: formulation of the models adapted to the new alternatives.

4.3.Implementation of the optimization tools: implementation of the optimization tools with the new algorithms.

5.Improvement of optimization efficiency by symmetry: implementation of heuristic methods in order to take advantage of the symmetries presented in typical airplane structures.

5.1.Study of the symmetries typically presented in airplane structures: study of the symmetry characteristics in typical semimonocoque structures presented in airplanes.

5.2.Formulation of the symmetries in the optimization model: formulation of heuristic methods to take advantage of the symmetry patterns found in semimonocoque structures.

5.3. Implementation of symmetry advantages in implementation tools: implementation of the methodology formulated in the last task within the optimization algorithms.

6.Adaptation of the optimization tools to new models of device: adaptation of new models of vibration absorber device formulated in the peer project [2] to the optimization tools.

7.Practical case study: optimization of a new practical case in order to test the improvements developed in the last tasks.

7.1.Problem set up: choice of interest points, and device properties, computation of FRF matrix and set up of algorithm parameters for maximum efficiency.

7.2.Solution and analysis of results: solution of the problem with the optimization tools developed and analysis of the results achieved following physics and efficiency criteria.

15.2.1 Propose schedule for future work

Name	Work
Adaptation of optimization tools to frequency windows	3d
Study of the effects of GA parameters on efficiency	**15d**
Study of mutation ratio effect	5d
Study of reproduction ratio effect	5d
Study of elitism effect	5d
Optimization of real-size problems	**15d**
Preprocessor	5d
Solution and analysis of results	10d
Test new optimization algorithms	**32d**
Evaluation of the possible alternatives	5d
Formulation of the new models	7d
Implementation of the optimization processes	20d
Improvement of optimization efficiency by symmetry	**23d**
Study of the symmetries typicalli presented by airplane structures	3d
Formulation of the symmetries in the optimization model	5d
Implementation of symmetry advantages in the optimization tools	15d
Adaptation of the optimization tools to new models of devices	4d
Practical cases study	**19d**
Problem set up	4d
Solution and analysis of results	15d

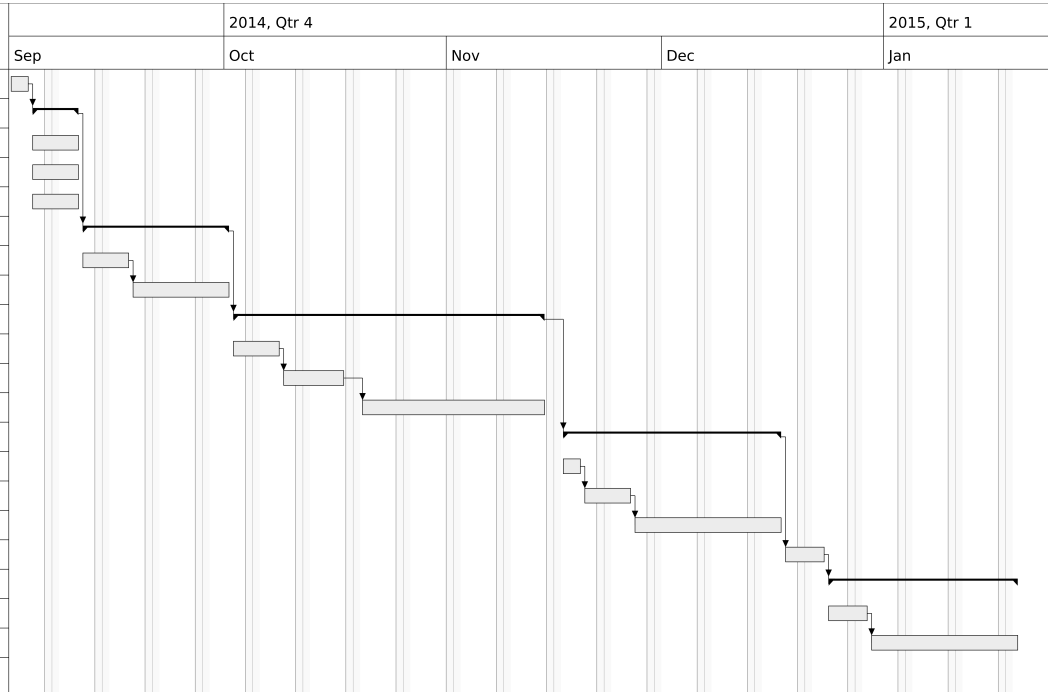

Figure 15.2: Gantt chart proposed to continue with the study.

16. Budget

The budget is presented in a separate document and it takes into account only the concepts related with the development of this study.

The total cost of this study comes to 13,923.05 €.

17. Economic viability

As it has been said before, this study has produced a tool capable of reducing the vibration levels by optimizing the positions and properties of the countermeasures. The improvement can also be performed determining a maximum value of accepted vibration and obtaining the countermeasure combination that accomplishes this level with the lowest total mass.

The cost of the process that leads to creating an operational tool to theoretically reduce mass in an airplane includes the budget presented in this study, as well as the one presented in the peer study developed by D.Sellés [2]. The total cost of both studies is 33033.05 €

This amount is tiny in comparison to the saving that weight reduction allows on, for example, a commercial airplane such as the Airbus A330. This airplane consumes an extra 0.15 kg of fuel per kilogram of extra weight in a flight of 4600 Km [21]. This section uses as a reference the orders of magnitude obtained in the optimization cases of the study, which suggest that the increase of weight of vibration countermeasures is roughly of a 0.5%. However a non optimal solution could have needed a greater increase of weight to achieve the same vibration reduction. Lets consider that the reduction in weight respect a non optimal solution is also around a 0.5%. In an Airbus A330 with an Operational Empty Weight of roughly 200 tons, this represents about 1000 kg of potential weight reduction, which leads to 150 kg of potential fuel mass reduction. With a density of approximately 0.8 kg/L, this implies a reduction of 186.567 L.

As an approximate value of Jet A fuel, the study uses the one found in the United States, which fluctuates around 0.566 €/L [22]. This, along with the estimated fuel reduction of 186.567 L for a 4600 km flight yields a reduction in cost of 0.02296 €/Km. Applying this measure to only one A330 airplane, the studies would be profitable after roughly 284 flights of 4600 Km. Since modern aircraft industries produce a large number of airplanes with the same structure, this number of flights could be spread amongst all airplanes of the same structure in which the optimization has been performed and, with slight modifications, to other similar models. The total cost reduction that the tool can yield highly depends on the amount of airplane units in which the countermeasures are applied, but the calculations show a great potential.

18. Conclusions

In general lines it can be said that the study has been a success because the initial requirements have been fulfilled and the scope of the study completed.

The aim of the project has been reached too, as proper optimization tools has been developed to solve the three main problems presented.

The tools developed in this study have been proved in practical cases achieving good results both in terms of vibration reduction and computational time. For the case placing only DVAs in a plate the reduction has been of 6.46dB, 3.36dB for the one placing DVAs in a fuselage structure and 9.95dB when adding also stiffeners.

Regarding the computational time, the time needed to obtain this reductions is at least $4.37 \cdot 10^{10}$ times lower than the one required to compute all the possible combinations in the case of DVAs in a plate, $4.04 \cdot 10^{10}$ times lower when they are placed in the fuselage and $1.48 \cdot 10^{16}$ times lower in the case of DVAs and stiffeners. When compared with the traditional alternative which recomputes the FEM model, the decrease of time is even greater. These results make it possible to optimize such problems. With this, it can be concluded that the initial requirements of the study are fulfilled.

The optimization processes are able to treat different vibrational countermeasures (DVAs, point masses and beams) as well as different types of all them in the same problem. This allows the engineer to deduce which is the cause of the vibration problem (lack of mass, lack of stiffness...) and to solve it optimally.

The results obtained show the power of the developed optimization processes despite their limitations. They open the path to future works with the aim of developing a fully functional tool able to optimize such problems, which can be really useful in aircraft, specially in advance level of design.

The product of this study not only presents engineering advantages but also a positive environmental impact. The optimization of vibration countermeasures reduce the weight of the aircraft without losing comfort, what is translated into a fuel consumption reduction. Besides, it positively affects directly to the acoustic pollution produced by the aircraft and the computational time savings achieved has a positive effect on the energy consumption.

Finally, it is remarkable that the study has been completely developed on schedule with just a few modification in the tasks planned which is for sure one of the causes of its

success.

19. Bibliography

[1] E.G. Tolosa González. Estudio de reducción de ruido mediante el uso de DVA. *Etseiat UPC*, TFC, 2012.

[2] D. Sellés Alseldà. Study of coupling of vibration countermeasures applied on airplane structure elements. *Etseiat UPC*, TFG, 2014.

[3] J. Schlegel. Fine-tuning vibro-acoustic countermeasures in a turboprop aircraft - SAE International.

[4] Y. Du, R. Burdisso, and E. Nikolaidis. Control of internal resonances in vibration isolators using passive and hybrid dynamic vibration absorbers. *Journal of Sound and Vibration*, 286(4-5):697–727, September 2005.

[5] S. Nastac and A. Leopa. Structural Optimization of Vibration Isolation Devices for High Performances. *International journal of systems applications, engineering & Development*, 2(2), 2008.

[6] C. Yang, D. Li, and L. Cheng. Dynamic vibration absorbers for vibration control within a frequency band. *Journal of Sound and Vibration*, 330(8):1582–1598, April 2011.

[7] Z. Qiu, B. Wang M. Shi, and Z.Xie. Genetic algorithm based active vibration control for a moving flexible smart beam driven by a pneumatic rod cylinder. *Journal of Sound and Vibration*, 331(10):2233–2256, May 2012.

[8] G.C. Marano and S. Sgobba. Optimal design criteria for isolation devices in vibration control. *intechopen.com*.

[9] W. Soedel. *Vibrations of shells and plates*. 2004.

[10] A. N. Letchford and H. P. Williams. *Model Building in Mathematical Programming*, volume 51. Fouth edition, October 2000.

[11] Linear programming. *http://en.wikipedia.org/wiki/Linear_programming*, May 2014.

[12] L.A. Wolsey. *Integer programming*. Willey-Intescience, 1998.

[13] Ibm ilog cplex optimization studio cplex user's manual. 2013.

[14] Integer programming. *http://en.wikipedia.org/wiki/Integer_programming*, May 2014.

[15] M. Obitko. Introduction to Genetic Algorithms. *http://www.obitko.com/tutorials/genetic-algorithms/search-space.php*, 1998.

[16] D. S. Johnson M.R. Garey. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[17] MathWorks - MATLAB and Simulink for Technical Computing. *http://www.mathworks.com/*, 31/05/14.

[18] IBM ILOG CPLEX Optimization Studio CPLEX Parameters Reference Manual. 2013.

[19] IBM ILOG CPLEX Optimization Studio CPLEX Get Started Manual. 2013.

[20] BBC News - Boeing checks 787 Dreamliners for wing cracks. *http://www.bbc.com/news/business-26492322*, 19/06/14.

[21] IATA Maintenance Cost Conference – October 2012. *https://www.iata.org/whatwedo/workgroups/Documents/MCC-2012-ATL/Day1/1150-1230-airbus-saving-fuel-team-sport.pdf*, 19/06/14.

[22] Jet Fuel - Daily Price - Commodity Prices - Price Charts, Data, and News - Index-Mundi. *http://www.indexmundi.com/commodities/?commodity=jet-fuel*, 19/06/14.